

EACL 2017

**15th Conference of the European Chapter of the
Association for Computational Linguistics**



Proceedings of Conference, Volume 1: Long Papers

April 3-7, 2017
Valencia, Spain

GOLD
SPONSORS



SILVER
SPONSORS



BRONZE
SPONSORS



©2017 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-945626-34-0

Preface: General Chair

Welcome to the EACL 2017, the 15th Conference of the European Chapter of the Association for Computational Linguistics! This is the largest ever EACL in terms of the number of papers being presented. We have a strong scientific program, including 14 workshops, six tutorials, a demos session, and a student research workshop. EACL received a record number of submissions this year, approximately 1,000 long and short papers combined, which reflects how broad and active our field is. We are also fortunate to have three excellent invited speakers: David Blei (University of Columbia), Devi Parikh (Virginia Tech), and Hinrich Schütze (LMU Munich). I hope that you will enjoy both the conference and Valencia.

I am deeply indebted to the Program Committee Chairs, Alexander Koller and Phil Blunsom, for their hard work. They put together a team of 27 area chairs who in turned assembled many reviewers and handled a large number of papers. The Workshop Chairs, Laura Rimmell and Richard Johansson, coordinated with the workshop chairs for ACL 2017 and EMNLP 2017 and succeeded in putting together an exciting and broad programme including 14 workshops. The student research workshop was organised by the student members of the EACL board — John Camilleri, Mariona Coll Ardanuy Uxoá Iñourrieta, and Florian Kunneman. With the help of Barbara Plank (Faculty advisor), they issued the call, organised a team of reviewers, assigned papers, coordinated and mediated among reviewers, and finally constructed a schedule consisting of 12 papers.

The Tutorial Chairs, Lucia Specia and Alexandre Klementiev, put together a very strong programme of six tutorials, which I hope many of us will attend. The publication chairs, Maria Liakata and Chris Biemann, have been short of amazing. They undertook the complex task of producing the conference proceedings and managed to make it seem easy, while being extremely thorough and paying attention to every detail. Chris Biemann deserves a double thank you for being Sponsorship Chair. Our demo chairs, Anselmo Peñas and André Martins, did a fantastic job selecting 30 demos for our demo session which I encourage you all to attend. I would also like to thank David Weir our publicity chair and the ACL business manager Priscilla Rassmussen, who knows more about our conferences than anyone else. Sincere thanks are due to the various sponsors for their generous contribution. I am grateful to all members of the EACL board for their advice and guidance, in particular to Lluís Márques and Walter Daelemans.

Last, but not least, this conference could not have taken place without the local organising committee who have worked tremendously hard to make EACL 2017 a success. The Local Chair, Paolo and Andrea Aldea from Grupo Pacifico, have brought together a fantastic local team and have dealt with many of the day-to-day tasks arising in organizing such a large conference expertly and efficiently.

I am always amazed by the dedication of our colleagues and their willingness to share knowledge and invest precious time in order to make our conferences a success. On that note, I would like to thank the authors who submitted their work to EACL and everyone else involved: area chairs, workshop organizers, tutorial presenters, reviewers, demo presenters, and participants of the conference.

Welcome to EACL 2017!

Mirella Lapata
General Chair

Preface: Programme Chairs

Welcome to the 15th Conference of the European Chapter of the Association for Computational Linguistics! In these proceedings you will find all the papers accepted for presentation at the conference in Valencia from the 3rd to the 7th of April 2017. The main conference program consists of both oral and poster presentations and also includes additional presentations of papers from the Transaction of the Association for Computational Linguistics (TACL), posters from the Student Research Workshop, and two demonstration sessions.

We received considerably more paper submissions than previous meetings of the EACL: 441 Long Papers and 502 Short Papers (excluding papers withdrawn or rejected for incorrect formatting). The Short Paper deadline was set after that for Long Papers and it is notable that we received more submissions of Short than Long papers. After the commendable reviewing efforts of our Program Committee we accepted 119 Long Papers, 78 as oral presentations and 41 posters, and 120 Short Papers, 47 orals and 73 posters. Overall the acceptance rates were 27% and 24% for the Long and Short Paper tracks respectively. The EACL 2017 programme also contained the oral presentations of four papers published in TACL.

It would not have been possible to produce such a high quality programme without the amazing effort and dedication of our Program Committee. We would like to thank all of those who served on the committee, which consisted of 27 Area Chairs and 612 Reviewers, drawn from a diverse range of fields and from both Europe and further afield. Each paper received at least three reviews. We selected the final programme based on the recommendations of the Area Chairs and reviewers, while aiming to ensure the representation of a wide variety of research areas. The Area Chairs were each asked to nominate candidate papers for the Outstanding Papers sessions, of which the Programme Chairs and General Chair selected three Long Papers and one Short Paper. These were allocated extra time in the programme for their oral presentations.

Following the precedent set at ACL 2016, we decided to allocate Long Paper and Short Paper oral presentations 20 minute and 15 minute slots respectively, including time for questions and changing speakers. While this shorter scheduling requires presenters to be more concise in their presentation, it allowed us to accommodate a larger program of talks in the space available at the venue.

In addition to the main conference programme, a Student Research Workshop was held which selected 12 papers for presentation as posters, and two demonstration sessions were held during the evening poster sessions. We are particularly grateful to our three distinguished invited speakers, Devi Parikh (Georgia Tech), David Blei (Columbia University), and Hinrich Schütze (LMU Munich). They represent the amazing diversity of contemporary research being conducted across Computational Linguistics, Artificial Intelligence, and Machine Learning.

In total the programme contains 126 talks and 126 posters, making this the largest EACL conference by a considerable margin. Firstly this would not be possible without the authors who chose to submit their research papers for publication at EACL, and we thank them for choosing our conference. Obviously coordinating such a programme requires contributions from many people beyond the Programme Chairs. We would like to thank our Area Chairs who ensured the smooth running of the two reviewing cycles. We are also thankful for the support we received from the rest of the organising committee, including the Publication Chairs, Local Organisers, Workshop Chairs, Tutorial Chairs, Demo Chairs, the Handbook Chair, and the Student Research Workshop Chair, all listed in full later in the proceedings. We are also grateful for the technical support received from the START team. We would like to thank the Programme Chairs for ACL 2016, Katrin Erk and Noah Smith, who generously provided many insights and tips from their own experience to help us avoid pitfalls and ensure the smooth running of the reviewing process. Finally, we are thankful to have been blessed with an exceptionally calm and organised General Chair in Mirella Lapata, who ensured the smooth running of the organising process and the ultimate success of

this conference.

We hope you enjoy EACL 2017 in Valencia!

Phil Blunsom and Alexander Koller
EACL 2017 Programme Chairs

Organisers

General Chair:

Mirella Lapata, University of Edinburgh

Program Chairs:

Phil Blunsom, University of Oxford

Alexander Koller, University of Saarbrücken

Local Organising Committee:

Paolo Rosso (Chair), PRHLT, Universitat Politècnica de València

Francisco Casacuberta (Co-chair), PRHLT, Universitat Politècnica de València

Jon Ander Gómez (Co-chair), PRHLT, Politècnica de València

Publication Chairs:

Maria Liakata, University of Warwick

Chris Biemann, University of Hamburg

Workshop Chairs:

Laura Rimell, University of Cambridge

Richard Johansson, University of Gothenberg

Tutorial Chairs:

Alex Klementiev, Amazon Berlin

Lucia Specia, University of Sheffield

Demo Chairs:

Anselmo Peñas, UNED, Madrid

André Martins, Unbabel Lda, Portugal

Student Research Workshop Chairs:

John J. Camilleri, University of Gothenburg

Mariona Coll Ardanuy, University of Göttingen

Uxo Iñurrieta, University of the Basque Country

Florian Kunneman, Radboud University

Student Research Workshop Faculty Advisor:

Barbara Plank, University of Groningen

Sponsorship Chairs:

Chris Biemann, University of Hamburg

Suzan Verberne, Leiden Institute of Advanced Computer Science

Publicity Chair:

David Weir, University of Sussex

Conference Handbook Chair:

Andreas Vlachos, University of Sheffield

Area Chairs:

Enrique Alfonseca, Nicholas Asher, Jason Baldridge, Alexandra Birch, Stephen Clark, Shay B. Cohen, Marcello Federico, Stefan L. Frank, Yoav Goldberg, Emiel Krahmer, Tom Kwiatkowski, Marie-Francine Moens, Malvina Nissim, Stephan Open, Miles Osborne, Rebecca J. Passonneau, Sebastian Riedel, Marcus Rohrbach, Andrew Rosenberg, Tatjana Scheffler, Hinrich Schütze, Gabriel Skantze, Mark Stevenson, Stephanie Strassel, Andreas Vlachos, Feiyu Xu, François Yvon

Reviewers:

Stergos Afantenos, Željko Agić, Alan Akbik, Nikolaos Aletras, Jan Alexandersson, Afra Alishahi, Tamer Alkhouli, Miltiadis Allamanis, Alexandre Allauzen, Carlos Alzate, Hadi Amiri, Waleed Ammar, Nicholas Andrews, Ion Androutsopoulos, Yoav Artzi, Isabelle Augenstein, Harald Baayen, Dzmitry Bahdanau, JinYeong Bak, Alexandra Balahur, Timothy Baldwin, Borja Balle, Miguel Ballesteros, David Bamman, Mohit Bansal, Daniel Bauer, Timo Baumann, Beata Beigman Klebanov, Núria Bel, Islam Beltagy, Anja Belz, Emily M. Bender, Andrew Bennett, Adrian Benton, Anton Benz, Jonathan Berant, Christina Bergmann, Laurent Besacier, Archana Bhatia, Yonatan Bisk, Johannes Bjerva, Frédéric Blain, Roi Blanco, Eduardo Blanco, Nate Blaylock, Nikolay Bogoychev, Bernd Bohnet, Gemma Boleda, Danushka Bollegala, Claire Bonial, Kalina Bontcheva, Johan Bos, Matko Bosnjak, Johan Boye, Chris Brew, Julian Brooke, Harm Brouwer, Elia Bruni, Christian Buck, Paul Buitelaar, José G. C. de Souza, Elena Cabrio, Deng Cai, Nicoletta Calzolari, Nick Campbell, Fabienne Cap, Xavier Carreras, Francisco Casacuberta, Daniel Cer, Mauro Cettolo, Nathanael Chambers, Kai-Wei Chang, Angel Chang, Rajen Chatterjee, Wanxiang Che, Danqi Chen, Yun-Nung Chen, Chen Chen, Boxing Chen, Hsin-Hsi Chen, Colin Cherry, Jackie Chi Kit Cheung, David Chiang, Christian Chiarcos, Do Kook Choe, Eunsol Choi, Monojit Choudhury, Christos Christodoulopoulos, Grzegorz Chrupała, Jennifer Chu-Carroll, Tagyoung Chung, Stephane Clinchant, Trevor Cohn, Nigel Collier, Michael Collins, John Conroy, Bonaventura Coppola, Ryan Cotterell, Danilo Croce, Heriberto Cuayahuitl, Walter Daelemans, Marina Danilevsky, Pradipto Das, Adrià de Gispert, Daniël de Kok, Gerard de Melo, Thierry Declerck, Marco Del Tredici, Estelle Delpesch, Vera Demberg, Thomas Demeester, Pascal Denis, Michael Denkowski, Tejaswini Deoskar, Leon Derczynski, Nina Dethlefs, Ann Devitt, Giuseppe Di Fabrizio, Mona Diab, Georgiana Dinu, Simon Dobnik, A. Seza Doğruöz, Markus Dreyer, Lan Du, Jason Duncan, Jesse Dunietz, Nadir Durrani, Jens Edlund, Koji Eguchi, Kathrin Eichler, Vladimir Eidelman, Michael Elhadad, Desmond Elliott, Micha Elsner, Ramy Eskander, Allyson Ettinger, Federico Fancellu, M. Amin Farajian, Geli Fei, Anna Feldman, Yansong Feng, Raquel Fernandez, Olivier Ferret, Katja Filippova, Andrew Finch, Nicholas FitzGerald, Antske Fokkens, José A. R. Fonolosa, Mikel Forcada, Martin Forst, George Foster, Jennifer Foster, Stella Frank, Anette Frank, Michael Franke, Dayne Freitag, Daniel Fried, Annemarie Friedrich, Hagen Fuerstenau, Alona Fyshe, Michel Galley, Michael Gamon, Kuzman Ganchev, Miguel A. García-Cumbreras, Claire Gardent, Matt Gardner, Milica Gasic, Albert Gatt, Eric Gaussier, Kallirroi Georgila, Kripabandhu Ghosh, Dafydd Gibbon, Daniel Gildea, Kevin Gimpel, Filip Ginter, Jonathan Ginzburg, Roxana Girju, Dimitra Gkatzia, Goran Glavaš, Yoav Goldberg, Dan Goldwasser, Juan Carlos Gomez, Kyle Gorman, Parantapa Goswami, Amit Goyal, Joao Graca, Yvette Graham, Mark Granroth-

Wilding, Ralph Grishman, Liane Guillou, Weiwei Guo, Joakim Gustafson, Nizar Habash, Ben Hachey, Barry Haddow, Gholamreza Haffari, Masato Hagiwara, Udo Hahn, Dilek Hakkani-Tur, John Hale, Bo Han, Sanda Harabagiu, Kazuma Hashimoto, Helen Hastie, Claudia Hauff, Daqing He, Yifan He, Luheng He, Kenneth Heafield, Sebastian Hellmann, Oliver Hellwig, Matthew Henderson, James Henderson, Lisa Anne Hendricks, Leonhard Hennig, Aurélie Herbelot, Jack Hessel, Dirk Hovy, Christine Howes, Ruihong Huang, Fei Huang, Matthias Huck, Mans Hulden, Muhammad Humayoun, Jena D. Hwang, Nancy Ide, Iustina Ilisei, Kentaro Inui, Hitoshi Isahara, Mohit Iyyer, Shahab Jalalvand, Srinivasan Janarthanam, Yacine Jernite, Yangfeng Ji, Wenbin Jiang, Richard Johansson, Kenneth Joseph, Patrick Juola, Dan Jurafsky, Gerhard Jäger, Nobuhiro Kaji, Jaap Kamps, Katharina Kann, Simon Keizer, Frank Keller, Casey Kennington, Douwe Kiela, Yubin Kim, Svetlana Kiritchenko, Julia Kiseleva, Dietrich Klakow, Manfred Klenner, Alistair Knott, Philipp Koehn, Rik Koncel-Kedziorski, Grzegorz Kondrak, Ioannis Konstas, Stefan Kopp, Moshe Koppel, Selcuk Kopru, Parisa Kordjamshidi, Valia Kordoni, Bhushan Kotnis, Mikhail Kozhevnikov, Sebastian Krause, Jayant Krishnamurthy, Canasai Kruengkrai, Lun-Wei Ku, Marco Kuhlmann, Roland Kuhn, Jonathan K. Kummerfeld, Polina Kuznetsova, Vasileios Lampos, Gerassimos Lampouras, Shalom Lappin, Birger Larsen, Staffan Larsson, Jey Han Lau, Alon Lavie, Angeliki Lazaridou, Joseph Le Roux, Moontae Lee, Sungjin Lee, Kenton Lee, Els Lefever, Alessandro Lenci, Gregor Leusch, Roger Levy, Mike Lewis, Chen Li, Junyi Jessy Li, Fangtao Li, Qi Li, Yunyao Li, Jing Li, Xiao Ling, Tal Linzen, Christina Lioma, Pierre Lison, Fei Liu, Kang Liu, Yang Liu, Shujie Liu, Jing Liu, Qun Liu, Varvara Logacheva, Aurelio Lopez-Lopez, Bin Lu, Wei Lu, Andy Luecking, Michal Lukasik, Zhunchen Luo, Minh-Thang Luong, Pranava Swaroop Madhyastha, Walid Magdy, Mateusz Malinowski, Shervin Malmasi, Gideon Mann, Diego Marcheggiani, Daniel Marcu, Scott Martin, Patricio Martinez-Barco, Héctor Martínez Alonso, Prashant Mathur, Takuya Matsuzaki, Austin Matthews, Evgeny Matusov, Arne Mauser, Diana McCarthy, David McClosky, Ryan McDonald, Florian Metzger, Adam Meyers, Haitao Mi, Timothy Miller, Tristan Miller, Seyed Abolghasem Mirroshandel, Teruko Mitamura, Daichi Mochihashi, Saif Mohammad, Karo Moilanen, Manuel Montes, Taesun Moon, Roser Morante, Mathieu Morey, Alessandro Moschitti, Philippe Muller, Maria Nadejde, Masaaki Nagata, Preslav Nakov, Courtney Napoles, Jason Naradowsky, Shashi Narayan, Tahira Naseem, Alexis Nasr, Borja Navarro, Roberto Navigli, Matteo Negri, Yael Netzer, Graham Neubig, Guenter Neumann, Mariana Neves, Hwee Tou Ng, Vincent Ng, Dong Nguyen, Vlad Niculae, Joakim Nivre, Pierre Nugues, Brendan O'Connor, Timothy O'Donnell, Kemal Oflazer, Jong-Hoon Oh, Alice Oh, Naoaki Okazaki, Tsuyoshi Okita, Constantin Orasan, Katja Ovchinnikova, Ulrike Pado, Muntsa Padró, Sebastian Padró, Alexis Palmer, Sinno Jialin Pan, Denis Paperno, Antonio Pareja Lora, Devi Parikh, Siddharth Patwardhan, Michael J. Paul, Ellie Pavlick, Bolette Pedersen, Hao Peng, Gerald Penn, Maciej Piasecki, Daniele Pighin, Mohammad Taher Pilehvar, Manfred Pinkal, Yuval Pinter, Emily Pitler, Paul Piwek, Barbara Plank, Massimo Poesio, Simone Paolo Ponzetto, Andrei Popescu-Belis, Maja Popović, François Portet, Alexandros Potamianos, Martin Potthast, Christopher Potts, Forough Poursabzi-Sangdeh, Daniel Preotiuc-Pietro, Stephen Pulman, Matthew Purver, James Pustejovsky, Xipeng Qiu, Guang Qiu, Afshin Rahimi, Altaf Rahman, Anita Ramm, Delip Rao, Ari Rappoport, Kyle Rawlins, Siva Reddy, Sravana Reddy, Ines Rehbein, Marek Rei, Roi Reichart, Ehud Reiter, David Reitter, Steffen Remus, Zhaochun Ren, Martin Riedl, Verena Rieser, Stefan Riezler, German Rigau, Brian Roark, Tim Rocktäschel, Horacio Rodriguez, Roland Roller, Stephen Roller, Carolyn Rose, Sara Rosenthal, Michael Roth, Sascha Rothe, Johann Roturier, Victoria Rubin, Markus Saers, Horacio Saggion, Benoît Sagot, Patrick Saint-Dizier, Hassan Sajjad, Avneesh Saluja, Rajhans Samdani, Mark Sammons, Anoop Sarkar, Felix Sasaki, Ryohei Sasano, Asad Sayeed, Carolina Scarton, David Schlangen, Natalie Schluter, Julian Schlöder, Helmut Schmid, Alexandra Schofield, William Schuler, Sabine Schulte im Walde, Roy Schwartz, H. Andrew Schwartz, Djamé Seddah, Frederique Segond, Satoshi Sekine, Rico Sennrich, Aliaksei Severyn, Kashif Shah, Serge Sharoff, Xiaodong Shi, Avirup Sil, Mario J. Silva, Khalil Sima'an, Kiril Simov, Sameer Singh, Kevin Small, Yan Song, Linfeng Song, Radu Soricut, Lucia Spe-

cia, Caroline Sporleder, Vivek Srikumar, Gabriel Stanovsky, Mark Steedman, Benno Stein, Pontus Stenetorp, Amanda Stent, Matthew Stone, Veselin Stoyanov, Karl Stratos, Kristina Striegnitz, Katsuhito Sudoh, Fei Sun, Weiwei Sun, Swabha Swayamdipta, Stan Szpakowicz, Felipe Sánchez-Martínez, Anders Søgaard, Hiroya Takamura, David Talbot, Partha Talukdar, Aleš Tamchyna, Jian Tang, Jiliang Tang, Makarand Tapaswi, Irina Temnikova, Joel Tetreault, Kapil Thadani, Mariët Theune, Jörg Tiedemann, Ivan Titov, Takenobu Tokunaga, Sara Tonelli, Fatemeh Torabi Asr, Kentaro Torisawa, Jennifer Tracey, Isabel Trancoso, Richard Tzong-Han Tsai, Reut Tsarfaty, Oren Tsur, Yoshimasa Tsuruoka, Marco Turchi, Oscar Täckström, Raghavendra Udupa, L. Alfonso Urena Lopez, Nicolas Usunier, Masao Utiyama, Benjamin Van Durme, Gertjan van Noord, Marten van Schijndel, Eva Maria Vecchi, Alakananda Vempala, Antoine Venant, Subhashini Venugopalan, Noortje Venuizen, Suzan Verberne, Yannick Versley, Laure Vieu, David Vilar, Rob Voigt, Martin Volk, Svitlana Volkova, Piek Vossen, Ivan Vulić, Ekaterina Vylomova, Marilyn Walker, Byron C. Wallace, Matthew Walter, Stephen Wan, Xiaojun Wan, Hsin-Min Wang, Wen Wang, Nigel Ward, Taro Watanabe, Andy Way, Bonnie Webber, Ingmar Weber, Julie Weeds, Albert Weichselbraun, Marion Weller-Di Marco, Dominikus Wetzel, Michael White, Michael Wiegand, Jason D. Williams, Shuly Wintner, Guillaume Wisniewski, Silke Witt-Ehsani, Kam-Fai Wong, Ji Wu, Hua Wu, Stephen Wu, Dekai Wu, Sander Wubben, Chunyang Xiao, Chenyan Xiong, Ruifeng Xu, Diyi Yang, Grace Hui Yang, Weiwei Yang, Yi Yang, Roman Yangarber, Mark Yatskar, Wenpeng Yin, Naoki Yoshinaga, Steve Young, Kai Yu, Annie Zaenen, Wajdi Zaghouni, Fabio Massimo Zanzotto, Alessandra Zarcone, Sina Zarrieß, Torsten Zesch, Luke Zettlemoyer, Deniz Zeyrek, Congle Zhang, Yue Zhang, Qi Zhang, Lei Zhang, Bing Zhao, Hai Zhao, Tiejun Zhao, Xiaodan Zhu, Heike Zinsmeister, Willem Zuidema, Özlem Çetinoğlu, Diarmuid Ó Séaghdha, Lilja Øvrelid, Jan Šnajder

Invited Speakers:

David Blei, Columbia University
Devi Parikh, Virginia Tech
Hinrich Schütze, LMU Munich

Invited Talk: David Blei

Title: Probabilistic Topic Models and User Behavior

Topic modeling algorithms analyze a document collection to estimate its latent thematic structure. However, many collections contain an additional type of data: how people use the documents. For example, readers click on articles in a newspaper website, scientists place articles in their personal libraries, and lawmakers vote on a collection of bills. Behavior data is essential both for making predictions about users (such as for a recommendation system) and for understanding how a collection and its users are organized.

I will review the basics of topic modeling and describe our recent research on collaborative topic models, models that simultaneously analyze a collection of texts and its corresponding user behavior. We studied collaborative topic models on 80,000 scientists' libraries from Mendeley and 100,000 users' click data from the arXiv. Collaborative topic models enable interpretable recommendation systems, capturing scientists' preferences and pointing them to articles of interest. Further, these models can organize the articles according to the discovered patterns of readership. For example, we can identify articles that are important within a field and articles that transcend disciplinary boundaries.

Biography:

David Blei is a Professor of Statistics and Computer Science at Columbia University, and a member of the Columbia Data Science Institute. His research is in statistical machine learning, involving probabilistic topic models, Bayesian nonparametric methods, and approximate posterior inference algorithms for massive data. He works on a variety of applications, including text, images, music, social networks, user behavior, and scientific data. David has received several awards for his research, including a Sloan Fellowship (2010), Office of Naval Research Young Investigator Award (2011), Presidential Early Career Award for Scientists and Engineers (2011), Blavatnik Faculty Award (2013), and ACM-Infosys Foundation Award (2013). He is a fellow of the ACM.

Invited Talk: Devi Parikh

Title: Words, Pictures, and Common Sense

Wouldn't it be nice if machines could understand content in images and communicate this understanding as effectively as humans? Such technology would be immensely powerful, be it for aiding a visually-impaired user navigate a world built by the sighted, assisting an analyst in extracting relevant information from a surveillance feed, educating a child playing a game on a touch screen, providing information to a spectator at an art gallery, or interacting with a robot. As computer vision and natural language processing techniques are maturing, we are closer to achieving this dream than we have ever been.

Visual Question Answering (VQA) is one step in this direction. Given an image and a natural language question about the image (e.g., "What kind of store is this?", "How many people are waiting in the queue?", "Is it safe to cross the street?"), the machine's task is to automatically produce an accurate natural language answer ("bakery", "5", "Yes"). In this talk, I will present our dataset, some neural models, and open research questions in free-form and open-ended Visual Question Answering (VQA). I will also show a teaser about the next step moving forward: Visual Dialog. Instead of answering individual questions about an image in isolation, can we build machines that can hold a sequential natural language conversation with humans about visual content?

While machines are getting better at superficially connecting words to pictures, interacting with them quickly reveals that they lack a certain common sense about the world we live in. Common sense is a key ingredient in building intelligent machines that make "human-like" decisions when performing tasks – be it automatically answering natural language questions, or understanding images and videos. How can machines learn this common sense? While some of this knowledge is explicitly stated in human-generated text (books, articles, blogs, etc.), much of this knowledge is unwritten. While unwritten, it is not unseen! The visual world around us is full of structure bound by commonsense laws. But machines today cannot learn common sense directly by observing our visual world because they cannot accurately perform detailed visual recognition in images and videos. We argue that one solution is to give up on photorealism. We propose to leverage abstract scenes – cartoon scenes made from clip art by crowd sourced humans – to teach our machines common sense. I will demonstrate how knowledge learnt from this abstract world can be used to solve commonsense textual tasks.

Biography:

Devi Parikh is an Assistant Professor in the School of Interactive Computing at Georgia Tech, and a Visiting Researcher at Facebook AI Research (FAIR). From 2013 to 2016, she was an Assistant Professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. From 2009 to 2012, she was a Research Assistant Professor at Toyota Technological Institute at Chicago (TTIC), an academic computer science institute affiliated with University of Chicago. She has held visiting positions at Cornell University, University of Texas at Austin, Microsoft Research, MIT, and Carnegie Mellon University. She received her M.S. and Ph.D. degrees from the Electrical and Computer Engineering department at Carnegie Mellon University in 2007 and 2009 respectively. She received her B.S. in Electrical and Computer Engineering from Rowan University in 2005. Her research interests include computer vision and AI in general and visual recognition problems in particular. Her recent work involves exploring problems at the intersection of vision and language, and leveraging human-machine collaboration for building smarter machines. She has also worked on other topics such as ensemble of classifiers, data fusion, inference in probabilistic models, 3D reassembly, barcode segmentation, computational photography, interactive computer vision, contextual reasoning, hierarchical representations of images, and human-debugging. She is a recipient of an NSF CAREER award, a Sloan Research Fellowship, an Office of Naval Research (ONR) Young Investigator Program (YIP) award, an Army Research Office (ARO) Young Investigator Program (YIP) award, an Allen Distinguished Investigator Award in Artificial Intelligence from the Paul G. Allen Family Foundation, four Google Faculty Research Awards,

an Amazon Academic Research Award, an Outstanding New Assistant Professor award from the College of Engineering at Virginia Tech, a Rowan University Medal of Excellence for Alumni Achievement, Rowan University's 40 under 40 recognition, and a Marr Best Paper Prize awarded at the International Conference on Computer Vision (ICCV).

Invited Talk: Hinrich Schütze

Title: Don't cram two completely different meanings into a single !&??@#^\$% vector! Or should you?

It is tempting to interpret a high-dimensional embedding space cartographically, i.e., as a map each point of which represents a distinct identifiable meaning – just as cities and mountains on a real map represent distinct identifiable geographic locations. On this interpretation, ambiguous words pose a problem: how can two completely different meanings be in the same location? Instead of learning a single embedding for an ambiguous word, should we rather learn a different embedding for each of its senses (as has often been proposed)? In this talk, I will take a fresh look at this question, drawing on simulations with pseudowords, sentiment analysis experiments, psycholinguistics and – if time permits – lexicography.

Biography:

Hinrich Schütze is professor of computational linguistics and director of the Center for Information and Language Processing at LMU Munich. He received his PhD from Stanford University's Department of Linguistics in 1995 and worked on natural language processing and information retrieval technology at Xerox PARC, at several Silicon Valley startups and at Google 1995-2004 and 2008/9. He coauthored *Foundations of Statistical Natural Language Processing* (with Chris Manning) and *Introduction to Information Retrieval* (with Chris Manning and Prabhakar Raghavan). His research is motivated by a fundamental question that computational linguists face today: Is domain knowledge about language dispensable (as many in deep learning seem to believe) or can linguistics and statistical NLP learn and benefit from each other?

Table of Contents

<i>Gated End-to-End Memory Networks</i>	
Fei Liu and Julien Perez	1
<i>Neural Tree Indexers for Text Understanding</i>	
Tsendsuren Munkhdalai and Hong Yu	11
<i>Exploring Different Dimensions of Attention for Uncertainty Detection</i>	
Heike Adel and Hinrich Schütze	22
<i>Classifying Illegal Activities on Tor Network Based on Web Textual Contents</i>	
Mhd Wesam Al Nabki, Eduardo Fidalgo, Enrique Alegre and Ivan de Paz	35
<i>When is multitask learning effective? Semantic sequence prediction under varying data conditions</i>	
Héctor Martínez Alonso and Barbara Plank	44
<i>Learning Compositionality Functions on Word Embeddings for Modelling Attribute Meaning in Adjective-Noun Phrases</i>	
Matthias Hartung, Fabian Kaupmann, Soufian Jebbara and Philipp Cimiano	54
<i>Hypernyms under Siege: Linguistically-motivated Artillery for Hypernymy Detection</i>	
Vered Shwartz, Enrico Santus and Dominik Schlechtweg	65
<i>Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network</i>	
Kim Anh Nguyen, Sabine Schulte im Walde and Ngoc Thang Vu	76
<i>Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation</i>	
Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone Paolo Ponzetto and Chris Biemann	86
<i>Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison</i>	
Alessandro Raganato, Jose Camacho-Collados and Roberto Navigli	99
<i>Which is the Effective Way for Gaokao: Information Retrieval or Neural Networks?</i>	
Shangmin Guo, Xiangrong Zeng, Shizhu He, Kang Liu and Jun Zhao	111
<i>If You Can't Beat Them Join Them: Handcrafted Features Complement Neural Nets for Non-Factoid Answer Reranking</i>	
Dasha Bogdanova, Jennifer Foster, Daria Dzendzik and Qun Liu	121
<i>Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks</i>	
Rajarshi Das, Arvind Neelakantan, David Belanger and Andrew McCallum	132
<i>Recognizing Mentions of Adverse Drug Reaction in Social Media Using Knowledge-Infused Recurrent Models</i>	
Gabriel Stanovsky, Daniel Gruhl and Pablo Mendes	142
<i>Multitask Learning for Mental Health Conditions with Limited Social Media Data</i>	
Adrian Benton, Margaret Mitchell and Dirk Hovy	152
<i>Evaluation by Association: A Systematic Study of Quantitative Word Association Evaluation</i>	
Ivan Vulić, Douwe Kiela and Anna Korhonen	163

<i>Computational Argumentation Quality Assessment in Natural Language</i>	
Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst and Benno Stein	176
<i>A method for in-depth comparative evaluation: How (dis)similar are outputs of pos taggers, dependency parsers and coreference resolvers really?</i>	
Don Tuggener	188
<i>Re-evaluating Automatic Metrics for Image Captioning</i>	
Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis and Erkut Erdem	199
<i>Integrating Meaning into Quality Evaluation of Machine Translation</i>	
Osman Baskaya, Eray Yildiz, Doruk Tunaoglu, Mustafa Tolga Eren and A. Seza Dogruoz	210
<i>Cross-Lingual Dependency Parsing with Late Decoding for Truly Low-Resource Languages</i>	
Michael Schlichtkrull and Anders Søgaard	219
<i>Parsing Universal Dependencies without training</i>	
Héctor Martínez Alonso, Željko Agić, Barbara Plank and Anders Søgaard	229
<i>Delexicalized Word Embeddings for Cross-lingual Dependency Parsing</i>	
Mathieu Dehouck and Pascal Denis	240
<i>Stance Classification of Context-Dependent Claims</i>	
Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha and Noam Slonim	250
<i>Exploring the Impact of Pragmatic Phenomena on Irony Detection in Tweets: A Multilingual Corpus Study</i>	
Jihen Karoui, Benamara Farah, Véronique Moriceau, Viviana Patti, Cristina Bosco and Nathalie Aussenac-Gilles	261
<i>A Multi-View Sentiment Corpus</i>	
Debora Nozza, Elisabetta Fersini and Enza Messina	272
<i>A Systematic Study of Neural Discourse Models for Implicit Discourse Relation</i>	
Attapol Rutherford, Vera Demberg and Nianwen Xue	280
<i>Cross-lingual RST Discourse Parsing</i>	
Chloé Braud, Maximin Coavoux and Anders Søgaard	291
<i>Dialog state tracking, a machine reading approach using Memory Network</i>	
Julien Perez and Fei Liu	304
<i>Sentence Segmentation in Narrative Transcripts from Neuropsychological Tests using Recurrent Convolutional Neural Networks</i>	
Marcos Treviso, Christopher Shulby and Sandra Aluísio	314
<i>Joint, Incremental Disfluency Detection and Utterance Segmentation from Speech</i>	
Julian Hough and David Schlangen	325
<i>From Segmentation to Analyses: a Probabilistic Model for Unsupervised Morphology Induction</i>	
Toms Bergmanis and Sharon Goldwater	336
<i>Creating POS Tagging and Dependency Parsing Experts via Topic Modeling</i>	
Atreyee Mukherjee, Sandra Kübler and Matthias Scheutz	346

<i>Universal Dependencies and Morphology for Hungarian - and on the Price of Universality</i> Veronika Vincze, Katalin Simkó, Zsolt Szántó and Richárd Farkas	355
<i>Addressing the Data Sparsity Issue in Neural AMR Parsing</i> Xiaochang Peng, Chuan Wang, Daniel Gildea and Nianwen Xue	365
<i>Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model</i> Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra and Sachindra Joshi	375
<i>Enumeration of Extractive Oracle Summaries</i> Tsutomu Hirao, Masaaki Nishino, Jun Suzuki and Masaaki Nagata	385
<i>Neural Semantic Encoders</i> Tsendsuren Munkhdalai and Hong Yu	396
<i>Efficient Benchmarking of NLP APIs using Multi-armed Bandits</i> Gholamreza Haffari, Tuan Dung Tran and Mark Carman	407
<i>Character-Word LSTM Language Models</i> Lyan Verwimp, Joris Pelemans, Hugo Van hamme and Patrick Wambacq	416
<i>A Hierarchical Neural Model for Learning Sequences of Dialogue Acts</i> Quan Hung Tran, Ingrid Zukerman and Gholamreza Haffari	427
<i>A Network-based End-to-End Trainable Task-oriented Dialogue System</i> Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes and Steve Young	437
<i>May I take your order? A Neural Model for Extracting Structured Information from Conversations</i> Baolin Peng, Michael Seltzer, Y.C. Ju, Geoffrey Zweig and Kam-Fai Wong	449
<i>A Two-stage Sieve Approach for Quote Attribution</i> Grace Muzny, Michael Fang, Angel Chang and Dan Jurafsky	459
<i>Out-of-domain FrameNet Semantic Role Labeling</i> Silvana Hartmann, Ilia Kuznetsov, Teresa Martin and Iryna Gurevych	470
<i>TDParse: Multi-target-specific sentiment recognition on Twitter</i> Bo Wang, Maria Liakata, Arkaitz Zubiaga and Rob Procter	482
<i>Annotating Derivations: A New Evaluation Strategy and Dataset for Algebra Word Problems</i> Shyam Upadhyay and Ming-Wei Chang	493
<i>An Extensive Empirical Evaluation of Character-Based Morphological Tagging for 14 Languages</i> Georg Heigold, Guenter Neumann and Josef van Genabith	504
<i>Neural Multi-Source Morphological Reinflection</i> Katharina Kann, Ryan Cotterell and Hinrich Schütze	513
<i>Online Automatic Post-editing for MT in a Multi-Domain Translation Environment</i> Rajen Chatterjee, Gebremedhen Gebremelak, Matteo Negri and Marco Turchi	524
<i>An Incremental Parser for Abstract Meaning Representation</i> Marco Damonte, Shay B. Cohen and Giorgio Satta	535

<i>Integrated Learning of Dialog Strategies and Semantic Parsing</i>	
Aishwarya Padmakumar, Jesse Thomason and Raymond J. Mooney	546
<i>Unsupervised AMR-Dependency Parse Alignment</i>	
Wei-Te Chen and Martha Palmer	557
<i>Improving Chinese Semantic Role Labeling using High-quality Surface and Deep Case Frames</i>	
Gongye Jin, Daisuke Kawahara and Sadao Kurohashi	567
<i>Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities</i>	
Yadollah Yaghoobzadeh and Hinrich Schütze	577
<i>The ContrastMedium Algorithm: Taxonomy Induction From Noisy Knowledge Graphs With Just A Few Links</i>	
Stefano Faralli, Alexander Panchenko, Chris Biemann and Simone Paolo Ponzetto	589
<i>Probabilistic Inference for Cold Start Knowledge Base Population with Prior World Knowledge</i>	
Bonan Min, Marjorie Freedman and Talya Meltzer	600
<i>Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema</i>	
Patrick Verga, Arvind Neelakantan and Andrew McCallum	612
<i>Learning to Generate Product Reviews from Attributes</i>	
Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou and Ke Xu	622
<i>Learning to generate one-sentence biographies from Wikidata</i>	
Andrew Chisholm, Will Radford and Ben Hachey	632
<i>Transition-Based Deep Input Linearization</i>	
Ratish Puduppully, Yue Zhang and Manish Shrivastava	642
<i>Generating flexible proper name references in text: Data, models and evaluation</i>	
Thiago Castro Ferreira, Emiel Krahmer and Sander Wubben	654
<i>Dependency Parsing as Head Selection</i>	
Xingxing Zhang, Jianpeng Cheng and Mirella Lapata	664
<i>Tackling Error Propagation through Reinforcement Learning: A Case of Greedy Dependency Parsing</i>	
Minh Le and Antske Fokkens	676
<i>Noisy-context surprisal as a human sentence processing cost model</i>	
Richard Futrell and Roger Levy	687
<i>Task-Specific Attentive Pooling of Phrase Alignments Contributes to Sentence Matching</i>	
Wenpeng Yin and Hinrich Schütze	698
<i>On-demand Injection of Lexical Knowledge for Recognising Textual Entailment</i>	
Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao and Daisuke Bekki	709
<i>Learning to Predict Denotational Probabilities For Modeling Entailment</i>	
Alice Lai and Julia Hockenmaier	720
<i>A Societal Sentiment Analysis: Predicting the Values and Ethics of Individuals by Analysing Social Media Content</i>	
Tushar Maheshwari, Aishwarya N. Reganti, Samiksha Gupta, Anupam Jamatia, Upendra Kumar, Björn Gambäck and Amitava Das	730

<i>Argument Strength is in the Eye of the Beholder: Audience Effects in Persuasion</i> Stephanie Lukin, Pranav Anand, Marilyn Walker and Steve Whittaker	741
<i>A Language-independent and Compositional Model for Personality Trait Recognition from Short Texts</i> Fei Liu, Julien Perez and Scott Nowson	753
<i>A Strong Baseline for Learning Cross-Lingual Word Embeddings from Sentence Alignments</i> Omer Levy, Anders Søgaard and Yoav Goldberg	764
<i>Online Learning of Task-specific Word Representations with a Joint Biconvex Passive-Aggressive Algorithm</i> Pascal Denis and Liva Ralaivola	774
<i>Nonsymbolic Text Representation</i> Hinrich Schütze	784
<i>Fine-Grained Entity Type Classification by Jointly Learning Representations and Label Embeddings</i> Abhishek Abhishek, Ashish Anand and Amit Awekar	796
<i>Event extraction from Twitter using Non-Parametric Bayesian Mixture Model with Word Embeddings</i> Deyu Zhou, Xuan Zhang and Yulan He	807
<i>End-to-end Relation Extraction using Neural Networks and Markov Logic Networks</i> Sachin Pawar, Pushpak Bhattacharyya and Girish Palshikar	817
<i>Trust, but Verify! Better Entity Linking through Automatic Verification</i> Benjamin Heinzerling, Michael Strube and Chin-Yew Lin	827
<i>Named Entity Recognition in the Medical Domain with Constrained CRF Models</i> Charles Jochim and Lea Deleris	838
<i>Learning and Knowledge Transfer with Memory Networks for Machine Comprehension</i> Mohit Yadav, Lovekesh Vig and Gautam Shroff	849
<i>If No Media Were Allowed inside the Venue, Was Anybody Allowed?</i> Zahra Sarabi and Eduardo Blanco	859
<i>Metaheuristic Approaches to Lexical Substitution and Simplification</i> Sallam Abualhaija, Tristan Miller, Judith Eckle-Kohler, Iryna Gurevych and Karl-Heinz Zimmermann	869
<i>Paraphrasing Revisited with Neural Machine Translation</i> Jonathan Mallinson, Rico Sennrich and Mirella Lapata	880
<i>Multilingual Training of Crosslingual Word Embeddings</i> Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird and Trevor Cohn	893
<i>Building Lexical Vector Representations from Concept Definitions</i> Danilo Silva de Carvalho and Minh Le Nguyen	904
<i>ShotgunWSD: An unsupervised algorithm for global word sense disambiguation inspired by DNA sequencing</i> Andrei Butnaru, Radu Tudor Ionescu and Florentina Hristea	915
<i>LanideNN: Multilingual Language Identification on Text Stream</i> Tom Kocmi and Ondřej Bojar	926

<i>Cross-Lingual Word Embeddings for Low-Resource Language Modeling</i> Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird and Trevor Cohn	936
<i>Consistent Translation of Repeated Nouns using Syntactic and Semantic Cues</i> Xiao Pu, Laura Mascarell and Andrei Popescu-Belis	947
<i>Psycholinguistic Models of Sentence Processing Improve Sentence Readability Ranking</i> David M. Howcroft and Vera Demberg	957
<i>Web-Scale Language-Independent Cataloging of Noisy Product Listings for E-Commerce</i> Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabbri and Ankur Datta.....	968
<i>Recognizing Insufficiently Supported Arguments in Argumentative Essays</i> Christian Stab and Iryna Gurevych	979
<i>Distributed Document and Phrase Co-embeddings for Descriptive Clustering</i> Motoki Sato, Austin J. Brockmeier, Georgios Kontonatsios, Tingting Mu, John Y. Goulermas, Jun'ichi Tsujii and Sophia Ananiadou	990
<i>SMARTies: Sentiment Models for Arabic Target entities</i> Noura Farra and Kathy McKeown.....	1001
<i>Exploring Convolutional Neural Networks for Sentiment Analysis of Spanish tweets</i> Isabel Segura-Bedmar, Antonio Quiros and Paloma Martínez	1013
<i>Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: A Generative Approach to Sentiment Analysis</i> Amr Mousa and Björn Schuller	1022
<i>Large-scale Opinion Relation Extraction with Distantly Supervised Neural Network</i> Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun and Qi Zhang	1032
<i>Decoding with Finite-State Transducers on GPUs</i> Arturo Argueta and David Chiang.....	1043
<i>Learning to Translate in Real-time with Neural Machine Translation</i> Jiatao Gu, Graham Neubig, Kyunghyun Cho and Victor O.K. Li.....	1052
<i>A Multifaceted Evaluation of Neural versus Phrase-Based Machine Translation for 9 Language Direc- tions</i> Antonio Toral and Víctor M. Sánchez-Cartagena	1062
<i>Personalized Machine Translation: Preserving Original Author Traits</i> Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia and Shuly Wintner.....	1073
<i>Bilingual Lexicon Induction by Learning to Combine Word-Level and Character-Level Representations</i> Geert Heyman, Ivan Vulić and Marie-Francine Moens.....	1084
<i>Grouping business news stories based on salience of named entities</i> Llorenc Escoter, Lidia Pivovarova, Mian Du, Anisia Katinskaia and Roman Yangarber	1095
<i>Very Deep Convolutional Networks for Text Classification</i> Alexis Conneau, Holger Schwenk, Loïc Barrault and Yann Lecun	1106
<i>"PageRank" for Argument Relevance</i> Henning Wachsmuth, Benno Stein and Yamen Ajjour	1116

<i>Predicting Counselor Behaviors in Motivational Interviewing Encounters</i> Verónica Pérez-Rosas, Rada Mihalcea, Kenneth Resnicow, Satinder Singh, Lawrence Ann, Kathy J. Goggin and Delwyn Catley	1127
<i>Authorship Attribution Using Text Distortion</i> Efsthathios Stamatatos	1137
<i>Structured Learning for Temporal Relation Extraction from Clinical Records</i> Artuur Leeuwenberg and Marie-Francine Moens	1149
<i>Entity Extraction in Biomedical Corpora: An Approach to Evaluate Word Embedding Features with PSO based Feature Selection</i> Shweta Yadav, Asif Ekbal, Sriparna Saha and Pushpak Bhattacharyya	1158
<i>Distant Supervision for Relation Extraction beyond the Sentence Boundary</i> Chris Quirk and Hoifung Poon	1170
<i>Noise Mitigation for Neural Entity Typing and Relation Extraction</i> Yadollah Yaghoobzadeh, Heike Adel and Hinrich Schütze	1182
<i>Analyzing Semantic Change in Japanese Loanwords</i> Hiroya Takamura, Ryo Nagata and Yoshifumi Kawasaki	1194
<i>Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists</i> Gerhard Jäger, Johann-Mattis List and Pavel Sofroniev	1204
<i>A Multi-task Approach to Predict Likability of Books</i> Suraj Maharjan, John Arevalo, Manuel Montes, Fabio A. González and Thamar Solorio	1216
<i>A Data-Oriented Model of Literary Language</i> Andreas van Cranenburgh and Rens Bod	1227
<i>Aye or naw, whit dae ye hink? Scottish independence and linguistic identity on social media</i> Philippa Shoemark, Debnil Sur, Luke Shrimpton, Iain Murray and Sharon Goldwater	1238
<i>What Do Recurrent Neural Network Grammars Learn About Syntax?</i> Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig and Noah A. Smith	1248
<i>Incremental Discontinuous Phrase Structure Parsing with the GAP Transition</i> Maximin Coavoux and Benoit Crabbé	1258
<i>Neural Architectures for Fine-grained Entity Type Classification</i> Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui and Sebastian Riedel	1270

Conference Program

Wednesday, April 5, 2017

9:30–10:50 *Invited talk: David Blei*

10:50–11:20 *Coffee break*

Session 1A: Machine Learning

11:20–11:40 *Gated End-to-End Memory Networks*
Fei Liu and Julien Perez

11:40–12:00 *Neural Tree Indexers for Text Understanding*
Tsendsuren Munkhdalai and Hong Yu

12:00–12:20 *Exploring Different Dimensions of Attention for Uncertainty Detection*
Heike Adel and Hinrich Schütze

12:20–12:40 *Classifying Illegal Activities on Tor Network Based on Web Textual Contents*
Mhd Wesam Al Nabki, Eduardo Fidalgo, Enrique Alegre and Ivan de Paz

12:40–13:00 *When is multitask learning effective? Semantic sequence prediction under varying data conditions*
Héctor Martínez Alonso and Barbara Plank

Wednesday, April 5, 2017 (continued)

Session 1B: Lexical Semantics

- 11:20–11:40 *Learning Compositionality Functions on Word Embeddings for Modelling Attribute Meaning in Adjective-Noun Phrases*
Matthias Hartung, Fabian Kaupmann, Soufian Jebbara and Philipp Cimiano
- 11:40–12:00 *Hypernyms under Siege: Linguistically-motivated Artillery for Hypernymy Detection*
Vered Shwartz, Enrico Santus and Dominik Schlechtweg
- 12:00–12:20 *Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network*
Kim Anh Nguyen, Sabine Schulte im Walde and Ngoc Thang Vu
- 12:20–12:40 *Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation*
Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone Paolo Ponzetto and Chris Biemann
- 12:40–13:00 *Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison*
Alessandro Raganato, Jose Camacho-Collados and Roberto Navigli

Session 1C: Information Retrieval and Information Extraction

- 11:20–11:40 *Which is the Effective Way for Gaokao: Information Retrieval or Neural Networks?*
Shangmin Guo, Xiangrong Zeng, Shizhu He, Kang Liu and Jun Zhao
- 11:40–12:00 *If You Can't Beat Them Join Them: Handcrafted Features Complement Neural Nets for Non-Factoid Answer Reranking*
Dasha Bogdanova, Jennifer Foster, Daria Dzendzik and Qun Liu
- 12:00–12:20 *Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks*
Rajarshi Das, Arvind Neelakantan, David Belanger and Andrew McCallum
- 12:20–12:40 *Recognizing Mentions of Adverse Drug Reaction in Social Media Using Knowledge-Infused Recurrent Models*
Gabriel Stanovsky, Daniel Gruhl and Pablo Mendes
- 12:40–13:00 *Multitask Learning for Mental Health Conditions with Limited Social Media Data*
Adrian Benton, Margaret Mitchell and Dirk Hovy

Wednesday, April 5, 2017 (continued)

Session 1D: Evaluation

- 11:20–11:40 *Evaluation by Association: A Systematic Study of Quantitative Word Association Evaluation*
Ivan Vulić, Douwe Kiela and Anna Korhonen
- 11:40–12:00 *Computational Argumentation Quality Assessment in Natural Language*
Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst and Benno Stein
- 12:00–12:20 *A method for in-depth comparative evaluation: How (dis)similar are outputs of pos taggers, dependency parsers and coreference resolvers really?*
Don Tuggener
- 12:20–12:40 *Re-evaluating Automatic Metrics for Image Captioning*
Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis and Erkut Erdem
- 12:40–13:00 *Integrating Meaning into Quality Evaluation of Machine Translation*
Osman Baskaya, Eray Yildiz, Doruk Tunaoglu, Mustafa Tolga Eren and A. Seza Dogruoz

13:00–14:30 *Lunch*

Session 2A: Parsing 1

- 14:30–14:50 *Cross-Lingual Dependency Parsing with Late Decoding for Truly Low-Resource Languages*
Michael Schlichtkrull and Anders Søgaard
- 14:50–15:10 *Parsing Universal Dependencies without training*
Héctor Martínez Alonso, Željko Agić, Barbara Plank and Anders Søgaard
- 15:10–15:30 *Delexicalized Word Embeddings for Cross-lingual Dependency Parsing*
Mathieu Dehouck and Pascal Denis

Wednesday, April 5, 2017 (continued)

Session 2B: Social Media 1

- 14:30–14:50 *Stance Classification of Context-Dependent Claims*
Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha and Noam Slonim
- 14:50–15:10 *Exploring the Impact of Pragmatic Phenomena on Irony Detection in Tweets: A Multilingual Corpus Study*
Jihen Karoui, Benamara Farah, Véronique Moriceau, Viviana Patti, Cristina Bosco and Nathalie Aussenac-Gilles
- 15:10–15:30 *A Multi-View Sentiment Corpus*
Debora Nozza, Elisabetta Fersini and Enza Messina

Session 2C: Discourse and Dialogue

- 14:30–14:50 *A Systematic Study of Neural Discourse Models for Implicit Discourse Relation*
Attapol Rutherford, Vera Demberg and Nianwen Xue
- 14:50–15:10 *Cross-lingual RST Discourse Parsing*
Chloé Braud, Maximin Coavoux and Anders Søgaard
- 15:10–15:30 *Dialog state tracking, a machine reading approach using Memory Network*
Julien Perez and Fei Liu

Session 2D: Segmentation

- 14:30–14:50 *Sentence Segmentation in Narrative Transcripts from Neuropsychological Tests using Recurrent Convolutional Neural Networks*
Marcos Treviso, Christopher Shulby and Sandra Aluísio
- 14:50–15:10 *Joint, Incremental Disfluency Detection and Utterance Segmentation from Speech*
Julian Hough and David Schlangen
- 15:10–15:30 *From Segmentation to Analyses: a Probabilistic Model for Unsupervised Morphology Induction*
Toms Bergmanis and Sharon Goldwater
- 15:30–16:00** *Coffee break*

Wednesday, April 5, 2017 (continued)

16:00–17:15 *Session 3A: Syntax and Machine Learning (See Vol.2, SP)*

16:00–17:30 *Session 3B: Generation, Summarisation, and QA (See Vol.2, SP)*

16:00–17:30 *Session 3C: Semantics (See Vol.2, SP)*

16:00–17:15 *Session 3D: Morphology and Psycholinguistics (See Vol.2, SP)*

17:30–19:30 *Long Posters 1*

Long Posters 1

Creating POS Tagging and Dependency Parsing Experts via Topic Modeling
Atreyee Mukherjee, Sandra Kübler and Matthias Scheutz

Universal Dependencies and Morphology for Hungarian - and on the Price of Universality
Veronika Vincze, Katalin Simkó, Zsolt Szántó and Richárd Farkas

Addressing the Data Sparsity Issue in Neural AMR Parsing
Xiaochang Peng, Chuan Wang, Daniel Gildea and Nianwen Xue

Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model
Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra and Sachindra Joshi

Enumeration of Extractive Oracle Summaries
Tsutomu Hirao, Masaaki Nishino, Jun Suzuki and Masaaki Nagata

Neural Semantic Encoders
Tsendsuren Munkhdalai and Hong Yu

Efficient Benchmarking of NLP APIs using Multi-armed Bandits
Gholamreza Haffari, Tuan Dung Tran and Mark Carman

Wednesday, April 5, 2017 (continued)

Character-Word LSTM Language Models

Lyan Verwimp, Joris Pelemans, Hugo Van hamme and Patrick Wambacq

A Hierarchical Neural Model for Learning Sequences of Dialogue Acts

Quan Hung Tran, Ingrid Zukerman and Gholamreza Haffari

A Network-based End-to-End Trainable Task-oriented Dialogue System

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes and Steve Young

May I take your order? A Neural Model for Extracting Structured Information from Conversations

Baolin Peng, Michael Seltzer, Y.C. Ju, Geoffrey Zweig and Kam-Fai Wong

A Two-stage Sieve Approach for Quote Attribution

Grace Muzny, Michael Fang, Angel Chang and Dan Jurafsky

Out-of-domain FrameNet Semantic Role Labeling

Silvana Hartmann, Iliia Kuznetsov, Teresa Martin and Iryna Gurevych

TDParse: Multi-target-specific sentiment recognition on Twitter

Bo Wang, Maria Liakata, Arkaitz Zubiaga and Rob Procter

Annotating Derivations: A New Evaluation Strategy and Dataset for Algebra Word Problems

Shyam Upadhyay and Ming-Wei Chang

An Extensive Empirical Evaluation of Character-Based Morphological Tagging for 14 Languages

Georg Heigold, Guenter Neumann and Josef van Genabith

Neural Multi-Source Morphological Reinflection

Katharina Kann, Ryan Cotterell and Hinrich Schütze

Online Automatic Post-editing for MT in a Multi-Domain Translation Environment

Rajen Chatterjee, Gebremedhen Gebremelak, Matteo Negri and Marco Turchi

17:30–19:30 *Short Posters 1 (See Vol.2, SP)*

17.30–19.30 *Student Research Workshop (See Vol.4, SRW)*

Wednesday, April 5, 2017 (continued)

17:30–19:30 *Demos (See Vol.3, Demos)*

Thursday, April 6, 2017

9:30–10:50 *Invited talk: Devi Parikh*

10:50–11:20 *Coffee break*

Session 4A: TACL

11:20–11:40 *Encoding Prior Knowledge with Eigenword Embeddings*
Dominique Osborne, Shashi Narayan and Shay B. Cohen

11:40–12:00 *Shift-Reduce Constituent Parsing with Neural Lookahead Features*
Jiangming Liu and Yue Zhang

12:00–12:20 *Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies*
Tal Linzen, Emmanuel Dupoux, Yoav Goldberg

12:20–12:40 *Automatically Tagging Constructions of Causation and Their Slot-Fillers*
Jesse Dunietz, Lori Levin, Jaime Carbonell

Session 4B: Semantic Analysis

11:20–11:40 *An Incremental Parser for Abstract Meaning Representation*
Marco Damonte, Shay B. Cohen and Giorgio Satta

11:40–12:00 *Integrated Learning of Dialog Strategies and Semantic Parsing*
Aishwarya Padmakumar, Jesse Thomason and Raymond J. Mooney

12:00–12:20 *Unsupervised AMR-Dependency Parse Alignment*
Wei-Te Chen and Martha Palmer

12:20–12:40 *Improving Chinese Semantic Role Labeling using High-quality Surface and Deep Case Frames*
Gongye Jin, Daisuke Kawahara and Sadao Kurohashi

Thursday, April 6, 2017 (continued)

Session 4C: Knowledge Bases

- 11:20–11:40 *Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities*
Yadollah Yaghoobzadeh and Hinrich Schütze
- 11:40–12:00 *The ContrastMedium Algorithm: Taxonomy Induction From Noisy Knowledge Graphs With Just A Few Links*
Stefano Faralli, Alexander Panchenko, Chris Biemann and Simone Paolo Ponzetto
- 12:00–12:20 *Probabilistic Inference for Cold Start Knowledge Base Population with Prior World Knowledge*
Bonan Min, Marjorie Freedman and Talya Meltzer
- 12:20–12:40 *Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema*
Patrick Verga, Arvind Neelakantan and Andrew McCallum

Session 4D: Generation

- 11:20–11:40 *Learning to Generate Product Reviews from Attributes*
Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou and Ke Xu
- 11:40–12:00 *Learning to generate one-sentence biographies from Wikidata*
Andrew Chisholm, Will Radford and Ben Hachey
- 12:00–12:20 *Transition-Based Deep Input Linearization*
Ratish Puduppully, Yue Zhang and Manish Shrivastava
- 12:20–12:40 *Generating flexible proper name references in text: Data, models and evaluation*
Thiago Castro Ferreira, Emiel Kraemer and Sander Wubben

13:00–14:30 *Lunch*

Thursday, April 6, 2017 (continued)

Session 5A: Parsing 2 and Psycholinguistics

- 14:30–14:50 *Dependency Parsing as Head Selection*
Xingxing Zhang, Jianpeng Cheng and Mirella Lapata
- 14:50–15:10 *Tackling Error Propagation through Reinforcement Learning: A Case of Greedy Dependency Parsing*
Minh Le and Antske Fokkens
- 15:10–15:30 *Noisy-context surprisal as a human sentence processing cost model*
Richard Futrell and Roger Levy

Session 5B: Entailment

- 14:30–14:50 *Task-Specific Attentive Pooling of Phrase Alignments Contributes to Sentence Matching*
Wenpeng Yin and Hinrich Schütze
- 14:50–15:10 *On-demand Injection of Lexical Knowledge for Recognising Textual Entailment*
Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao and Daisuke Bekki
- 15:10–15:30 *Learning to Predict Denotational Probabilities For Modeling Entailment*
Alice Lai and Julia Hockenmaier

Session 5C: Social Media 2

- 14:30–14:50 *A Societal Sentiment Analysis: Predicting the Values and Ethics of Individuals by Analysing Social Media Content*
Tushar Maheshwari, Aishwarya N. Reganti, Samiksha Gupta, Anupam Jamatia, Upendra Kumar, Björn Gambäck and Amitava Das
- 14:50–15:10 *Argument Strength is in the Eye of the Beholder: Audience Effects in Persuasion*
Stephanie Lukin, Pranav Anand, Marilyn Walker and Steve Whittaker
- 15:10–15:30 *A Language-independent and Compositional Model for Personality Trait Recognition from Short Texts*
Fei Liu, Julien Perez and Scott Nowson

Thursday, April 6, 2017 (continued)

Session 5D: Word Representations

14:30–14:50 *A Strong Baseline for Learning Cross-Lingual Word Embeddings from Sentence Alignments*
Omer Levy, Anders Søgaard and Yoav Goldberg

14:50–15:10 *Online Learning of Task-specific Word Representations with a Joint Biconvex Passive-Aggressive Algorithm*
Pascal Denis and Liva Ralaivola

15:10–15:30 *Nonsymbolic Text Representation*
Hinrich Schütze

16:00–17:30 *Session 6A: Machine Translation (See Vol.2, SP)*

16:00–17:30 *Session 6B: Word Embeddings (See Vol.2, SP)*

16:00–17:30 *Session 6C: Document Analysis (See Vol.2, SP)*

16:00–17:15 *Session 6D: Dialogue (See Vol.2, SP)*

17:30–19:30 *Long Posters 2*

Thursday, April 6, 2017 (continued)

Long Posters 2

Fine-Grained Entity Type Classification by Jointly Learning Representations and Label Embeddings

Abhishek Abhishek, Ashish Anand and Amit Awekar

Event extraction from Twitter using Non-Parametric Bayesian Mixture Model with Word Embeddings

Deyu Zhou, Xuan Zhang and Yulan He

End-to-end Relation Extraction using Neural Networks and Markov Logic Networks

Sachin Pawar, Pushpak Bhattacharyya and Girish Palshikar

Trust, but Verify! Better Entity Linking through Automatic Verification

Benjamin Heinzerling, Michael Strube and Chin-Yew Lin

Named Entity Recognition in the Medical Domain with Constrained CRF Models

Charles Jochim and Lea Deleris

Learning and Knowledge Transfer with Memory Networks for Machine Comprehension

Mohit Yadav, Lovekesh Vig and Gautam Shroff

If No Media Were Allowed inside the Venue, Was Anybody Allowed?

Zahra Sarabi and Eduardo Blanco

Metaheuristic Approaches to Lexical Substitution and Simplification

Sallam Abualhaija, Tristan Miller, Judith Eckle-Kohler, Iryna Gurevych and Karl-Heinz Zimmermann

Paraphrasing Revisited with Neural Machine Translation

Jonathan Mallinson, Rico Sennrich and Mirella Lapata

Multilingual Training of Crosslingual Word Embeddings

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird and Trevor Cohn

Building Lexical Vector Representations from Concept Definitions

Danilo Silva de Carvalho and Minh Le Nguyen

ShotgunWSD: An unsupervised algorithm for global word sense disambiguation inspired by DNA sequencing

Andrei Butnaru, Radu Tudor Ionescu and Florentina Hristea

Thursday, April 6, 2017 (continued)

LanideNN: Multilingual Language Identification on Text Stream

Tom Kocmi and Ondřej Bojar

Cross-Lingual Word Embeddings for Low-Resource Language Modeling

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird and Trevor Cohn

Consistent Translation of Repeated Nouns using Syntactic and Semantic Cues

Xiao Pu, Laura Mascarell and Andrei Popescu-Belis

Psycholinguistic Models of Sentence Processing Improve Sentence Readability Ranking

David M. Howcroft and Vera Demberg

Web-Scale Language-Independent Cataloging of Noisy Product Listings for E-Commerce

Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabbrizio and Ankur Datta

Recognizing Insufficiently Supported Arguments in Argumentative Essays

Christian Stab and Iryna Gurevych

Distributed Document and Phrase Co-embeddings for Descriptive Clustering

Motoki Sato, Austin J. Brockmeier, Georgios Kononatsios, Tingting Mu, John Y. Goulermas, Jun'ichi Tsujii and Sophia Ananiadou

SMARTies: Sentiment Models for Arabic Target entities

Noura Farra and Kathy McKeown

Exploring Convolutional Neural Networks for Sentiment Analysis of Spanish tweets

Isabel Segura-Bedmar, Antonio Quiros and Paloma Martínez

Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: A Generative Approach to Sentiment Analysis

Amr Mousa and Björn Schuller

Large-scale Opinion Relation Extraction with Distantly Supervised Neural Network

Changzhi Sun, Yuanbin Wu, Man Lan, Shiliang Sun and Qi Zhang

17:30–19:30 *Short Posters 2 (See Vol.2, SP)*

17:30–19:30 *Demos (See Vol.3, Demos)*

Friday, April 7, 2017

9:30–10:50 *Invited talk: Hinrich Schütze*

10:50–11:20 *Coffee break*

Session 7A: Machine Translation and Multilinguality

11:20–11:40 *Decoding with Finite-State Transducers on GPUs*

Arturo Argueta and David Chiang

11:40–12:00 *Learning to Translate in Real-time with Neural Machine Translation*

Jiatao Gu, Graham Neubig, Kyunghyun Cho and Victor O.K. Li

12:00–12:20 *A Multifaceted Evaluation of Neural versus Phrase-Based Machine Translation for 9 Language Directions*

Antonio Toral and Víctor M. Sánchez-Cartagena

12:20–12:40 *Personalized Machine Translation: Preserving Original Author Traits*

Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia and Shuly Wintner

12:40–13:00 *Bilingual Lexicon Induction by Learning to Combine Word-Level and Character-Level Representations*

Geert Heyman, Ivan Vulić and Marie-Francine Moens

Friday, April 7, 2017 (continued)

Session 7B: Document Analysis

- 11:20–11:40 *Grouping business news stories based on salience of named entities*
Llorenç Escoter, Lidia Pivovarova, Mian Du, Anisia Katinskaia and Roman Yangarber
- 11:40–12:00 *Very Deep Convolutional Networks for Text Classification*
Alexis Conneau, Holger Schwenk, Loïc Barrault and Yann Lecun
- 12:00–12:20 *"PageRank" for Argument Relevance*
Henning Wachsmuth, Benno Stein and Yamen Ajjour
- 12:20–12:40 *Predicting Counselor Behaviors in Motivational Interviewing Encounters*
Verónica Pérez-Rosas, Rada Mihalcea, Kenneth Resnicow, Satinder Singh, Lawrence Ann, Kathy J. Goggin and Delwyn Catley
- 12:40–13:00 *Authorship Attribution Using Text Distortion*
Efstathios Stamatatos

Session 7C: Entity and Relation Extraction

- 11:20–11:40 *Structured Learning for Temporal Relation Extraction from Clinical Records*
Artuur Leeuwenberg and Marie-Francine Moens
- 11:40–12:00 *Entity Extraction in Biomedical Corpora: An Approach to Evaluate Word Embedding Features with PSO based Feature Selection*
Shweta Yadav, Asif Ekbal, Sriparna Saha and Pushpak Bhattacharyya
- 12:00–12:20 *Distant Supervision for Relation Extraction beyond the Sentence Boundary*
Chris Quirk and Hoifung Poon
- 12:20–12:40 *Noise Mitigation for Neural Entity Typing and Relation Extraction*
Yadollah Yaghoobzadeh, Heike Adel and Hinrich Schütze

Friday, April 7, 2017 (continued)

Session 7D: Historical and Literary Language

- 11:20–11:40 *Analyzing Semantic Change in Japanese Loanwords*
Hiroya Takamura, Ryo Nagata and Yoshifumi Kawasaki
- 11:40–12:00 *Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists*
Gerhard Jäger, Johann-Mattis List and Pavel Sofroniev
- 12:00–12:20 *A Multi-task Approach to Predict Likability of Books*
Suraj Maharjan, John Arevalo, Manuel Montes, Fabio A. González and Thamar Solorio
- 12:20–12:40 *A Data-Oriented Model of Literary Language*
Andreas van Cranenburgh and Rens Bod
- 12:40–13:00 *Aye or naw, whit dae ye hink? Scottish independence and linguistic identity on social media*
Philippa Shoemark, Debnil Sur, Luke Shrimpton, Iain Murray and Sharon Goldwater
- 13:00–14:30** *Lunch*
- 14:30–15:30** *Business Meeting*
- 15:30–16:00** *Coffee break*

Friday, April 7, 2017 (continued)

Session 8A: Outstanding Papers 1

16:00–16:25 *What Do Recurrent Neural Network Grammars Learn About Syntax?*
Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig and Noah A. Smith

16:25–16:50 *Best Short Paper (See Vol.2, SP)*

Session 8B: Outstanding Papers 2

16:00–16:25 *Incremental Discontinuous Phrase Structure Parsing with the GAP Transition*
Maximin Coavoux and Benoit Crabbé

16:25–16:50 *Neural Architectures for Fine-grained Entity Type Classification*
Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui and Sebastian Riedel

16:55–17:10 *Closing Session*

Gated End-to-End Memory Networks

Fei Liu ^{*†}

The University of Melbourne
Victoria, Australia

fliu3@student.unimelb.edu.au

Julien Perez [†]

Xerox Research Centre Europe
Grenoble, France

julien.perez@xrce.xerox.com

Abstract

Machine reading using differentiable reasoning models has recently shown remarkable progress. In this context, End-to-End trainable Memory Networks (MemN2N) have demonstrated promising performance on simple natural language based reasoning tasks such as factual reasoning and basic deduction. However, other tasks, namely multi-fact question-answering, positional reasoning or dialog related tasks, remain challenging particularly due to the necessity of more complex interactions between the memory and controller modules composing this family of models. In this paper, we introduce a novel end-to-end memory access regulation mechanism inspired by the current progress on the connection short-cutting principle in the field of computer vision. Concretely, we develop a Gated End-to-End trainable Memory Network architecture (GMemN2N). From the machine learning perspective, this new capability is learned in an end-to-end fashion without the use of any additional supervision signal which is, as far as our knowledge goes, the first of its kind. Our experiments show significant improvements on the most challenging tasks in the 20 bAbI dataset, without the use of any domain knowledge. Then, we show improvements on the Dialog bAbI tasks including the real human-bot conversation-based Dialog State Tracking Challenge (DSTC-2) dataset. On these two datasets, our model sets the new state of the art.

* Work carried out as an intern at XRCE

† Equal contribution

1 Introduction

Deeper Neural Network models are more difficult to train and recurrency tends to complicate this optimization problem (Srivastava et al., 2015b). While Deep Neural Network architectures have shown superior performance in numerous areas, such as image, speech recognition and more recently text, the complexity of optimizing such large and non-convex parameter sets remains a challenge. Indeed, the so-called vanishing/exploding gradient problem has been mainly addressed using: 1. algorithmic responses, e.g., normalized initialization strategies (LeCun et al., 1998; Glorot and Bengio, 2010); 2. architectural ones, e.g., intermediate normalization layers which facilitate the convergence of networks composed of tens of hidden layers (He et al., 2015; Saxe et al., 2014). Another problem of memory-enhanced neural models is the necessity of regulating memory access at the controller level. Memory access operations can be supervised (Kumar et al., 2016) and the number of times they are performed tends to be fixed apriori (Sukhbaatar et al., 2015), a design choice which tends to be based on the presumed degree of difficulty of the task in question. Inspired by the recent success of object recognition in the field of computer vision (Srivastava et al., 2015a; Srivastava et al., 2015b), we investigate the use of a *gating* mechanism in the context of End-to-End Memory Networks (MemN2N) (Sukhbaatar et al., 2015) in order to regulate the access to the memory blocks in a differentiable fashion. The formulation is realized by gated connections between the memory access layers and the controller stack of a MemN2N. As a result, the model is able to dynamically determine how and when to skip its memory-based reasoning process.

Roadmap: Section 2 reviews state-of-the-art Memory Network models, connection short-

cutting in neural networks and memory dynamics. In Section 3, we propose a differentiable gating mechanism in MemN2N. Section 4 and 5 present a set of experiments on the 20 bAbI reasoning tasks and the Dialog bAbI dataset. We report new state-of-the-art results on several of the most challenging tasks of the set, namely positional reasoning, 3-argument relation and the DSTC-2 task while maintaining equally competitive performance on the rest.

2 Related Work

This section starts with an introduction of the primary components of MemN2N. Then, we review two key elements relevant to this work, namely shortcut connections in neural networks and memory dynamics in such models.

2.1 End-to-End Memory Networks

The MemN2N architecture, introduced by Sukhbaatar et al. (2015), consists of two main components: supporting memories and final answer prediction. Supporting memories are in turn comprised of a set of input and output memory representations with memory cells. The input and output memory cells, denoted by \mathbf{m}_i and \mathbf{c}_i , are obtained by transforming the input context x_1, \dots, x_n (or stories) using two embedding matrices \mathbf{A} and \mathbf{C} (both of size $d \times |V|$ where d is the embedding size and $|V|$ the vocabulary size) such that $\mathbf{m}_i = \mathbf{A}\Phi(x_i)$ and $\mathbf{c}_i = \mathbf{C}\Phi(x_i)$ where $\Phi(\cdot)$ is a function that maps the input into a bag of dimension $|V|$. Similarly, the question q is encoded using another embedding matrix $\mathbf{B} \in \mathbb{R}^{d \times |V|}$, resulting in a question embedding $\mathbf{u} = \mathbf{B}\Phi(q)$. The input memories $\{\mathbf{m}_i\}$, together with the embedding of the question \mathbf{u} , are utilized to determine the relevance of each of the stories in the context, yielding a vector of attention weights

$$p_i = \text{softmax}(\mathbf{u}^\top \mathbf{m}_i) \quad (1)$$

where $\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}}$. Subsequently, the response \mathbf{o} from the output memory is constructed by the weighted sum:

$$\mathbf{o} = \sum_i p_i \mathbf{c}_i \quad (2)$$

For more difficult tasks demanding multiple supporting memories, the model can be extended

to include more than one set of input/output memories by stacking a number of memory layers. In this setting, each memory layer is named a hop and the $(k+1)$ th hop takes as input the output of the k th hop:

$$\mathbf{u}^{k+1} = \mathbf{o}^k + \mathbf{u}^k \quad (3)$$

Lastly, the final step, the prediction of the answer to the question q , is performed by

$$\hat{\mathbf{a}} = \text{softmax}(\mathbf{W}(\mathbf{o}^K + \mathbf{u}^K)) \quad (4)$$

where $\hat{\mathbf{a}}$ is the predicted answer distribution, $\mathbf{W} \in \mathbb{R}^{|V| \times d}$ is a parameter matrix for the model to learn and K the total number of hops.

2.2 Shortcut Connections

Shortcut connections have been studied from both the theoretical and practical point of view in the general context of neural network architectures (Bishop, 1995; Ripley, 2007). More recently Residual Networks (He et al., 2016) and Highway Networks (Srivastava et al., 2015a; Srivastava et al., 2015b) have been almost simultaneously proposed. While the former utilizes a residual calculus, the latter formulates a differentiable gateway mechanism as proposed in Long-Short Terms Memory Networks (Hochreiter and Schmidhuber, 1997) in order to cope with long-term dependency issues in the dataset in an end-to-end trainable manner. These two mechanisms were proposed as a structural solution to the so-called vanishing gradient problem by allowing the model to shortcut its layered transformation structure when necessary.

2.3 Memory Dynamics

The necessity of dynamically regulating the interaction between the so-called controller and the memory blocks of a Memory Network model has been studied in (Kumar et al., 2016; Xiong et al., 2016). In these works, the number of exchanges between the controller stack and the memory module of the network is either monitored in a hard supervised manner in the former or fixed a priori in the latter.

In this paper, we propose an end-to-end supervised model, with an automatically learned gating mechanism, to perform dynamic regulation of memory interaction. The next section presents the formulation of this new Gated End-to-End Memory Networks (GMemN2N). This contribution can be placed in parallel to the recent transition from Memory Networks with hard attention mechanism

(Weston et al., 2015) to MemN2N with attention values obtained by a softmax function and end-to-end supervised (Sukhbaatar et al., 2015).

3 Gated End-to-End Memory Network

In this section, the elements behind residual learning and highway neural models are given. Then, we introduce the proposed model of memory access gating in a MemN2N.

3.1 Highway and Residual Networks

Highway Networks, first introduced by Srivastava et al. (2015a), include a transform gate T and a carry gate C , allowing the network to learn how much information it should transform or carry to form the input to the next layer. Suppose the original network is a plain feed-forward neural network:

$$\mathbf{y} = H(\mathbf{x}) \quad (5)$$

where $H(\mathbf{x})$ is a non-linear transformation of its input \mathbf{x} . The generic form of Highway Networks is formulated as:

$$\mathbf{y} = H(\mathbf{x}) \odot T(\mathbf{x}) + \mathbf{x} \odot C(\mathbf{x}) \quad (6)$$

where the transform and carry gates, $T(\mathbf{x})$ and $C(\mathbf{x})$, are defined as non-linear transformation functions of the input \mathbf{x} and \odot the Hadamard product. As suggested in (Srivastava et al., 2015a; Srivastava et al., 2015b), we choose to focus, in the following of this paper, on a simplified version of Highway Networks where the carry gate is replaced by $1 - T(\mathbf{x})$:

$$\mathbf{y} = H(\mathbf{x}) \odot T(\mathbf{x}) + \mathbf{x} \odot (1 - T(\mathbf{x})) \quad (7)$$

where $T(\mathbf{x}) = \sigma(\mathbf{W}_T \mathbf{x} + \mathbf{b}_T)$ and σ is the sigmoid function. In fact, Residual Networks can be viewed as a special case of Highway Networks where both the transform and carry gates are substituted by the identity mapping function:

$$\mathbf{y} = H(\mathbf{x}) + \mathbf{x} \quad (8)$$

thereby forming a hard-wired shortcut connection \mathbf{x} .

3.2 Gated End-to-End Memory Networks

Arguably, Equation (3) can be considered as a form of residuality with \mathbf{o}^k working as the residual function and \mathbf{u}^k the shortcut connection. However, as discussed in (Srivastava et al., 2015b),

in contrast to the hard-wired skip connection in Residual Networks, one of the advantages of Highway Networks is the adaptive gating mechanism, capable of learning to dynamically control the information flow based on the current input. Therefore, we adopt the idea of the adaptive gating mechanism of Highway Networks and integrate it into MemN2N. The resulting model, named *Gated End-to-End Memory Networks* (GMemN2N) and illustrated in Figure 1, is capable of dynamically conditioning the memory reading operation on the controller state \mathbf{u}^k at each hop. Concretely, we reformulate Equation (3) into:

$$T^k(\mathbf{u}^k) = \sigma(\mathbf{W}_T^k \mathbf{u}^k + \mathbf{b}_T^k) \quad (9)$$

$$\mathbf{u}^{k+1} = \mathbf{o}^k \odot T^k(\mathbf{u}^k) + \mathbf{u}^k \odot (1 - T^k(\mathbf{u}^k)) \quad (10)$$

where \mathbf{W}_T^k and \mathbf{b}_T^k are the hop-specific parameter matrix and bias term for the k^{th} hop and $T^k(x)$ the transform gate for the k^{th} hop. Similar to the two weight tying schemes of the embedding matrices introduced in (Sukhbaatar et al., 2015), we also explore two types of constraints on \mathbf{W}_T^k and \mathbf{b}_T^k :

1. **Global:** all the weight matrices \mathbf{W}_T^k and bias terms \mathbf{b}_T^k are shared across different hops, i.e., $\mathbf{W}_T^1 = \mathbf{W}_T^2 = \dots = \mathbf{W}_T^K$ and $\mathbf{b}_T^1 = \mathbf{b}_T^2 = \dots = \mathbf{b}_T^K$.
2. **Hop-specific:** each hop has its specific weight matrix \mathbf{W}_T^k and bias term \mathbf{b}_T^k for $k \in [1, K]$ and they are optimized independently.

4 QA bAbI Experiments

In this section, we first describe the natural language reasoning dataset we use in our experiments. Then, the experimental setup is detailed. Lastly, we present the results and analyses.

4.1 Dataset and Data Preprocessing

The 20 bAbI tasks (Weston et al., 2016) have been employed for the experiments (using v1.2 of the dataset). In this synthetically generated dataset, a given QA task consists of a set of statements, followed by a question whose answer is typically a single word (in a few tasks, answers are a set of words). The answer is available to the model at training time but must be predicted at test time. The dataset consists of 20 different tasks with various emphases on different forms of reasoning. For each question, only a certain subset of the statements contains information needed for the answer, and the rest are essentially irrelevant distractors.

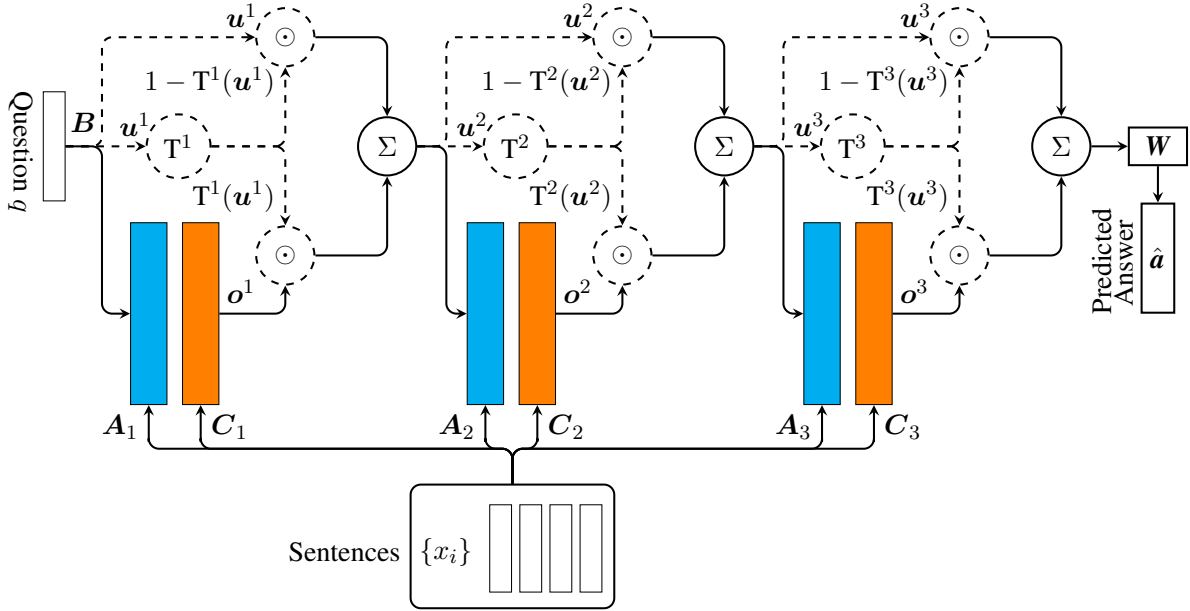


Figure 1: Illustration of the proposed GMemN2N model with 3 hops. Dashed lines indicate elements different from MemN2N (Sukhbaatar et al., 2015).

As in (Sukhbaatar et al., 2015), our model is fully end-to-end trained without any additional supervision other than the answers themselves. Formally, for one of the 20 QA tasks, we are given example problems, each having a set of I sentences $\{x_i\}$ (where $I \leq 320$), a question sentence q and answer a . Let the j^{th} word of sentence i be x_{ij} , represented by a one-hot vector of length $|V|$. The same representation is used for the question q and answer a . Two versions of the data are used, one that has 1,000 training problems per task and the other with 10,000 per task.

4.2 Training Details

As suggested in (Sukhbaatar et al., 2015), 10% of the bAbI training set was held-out to form a validation set for hyperparameter tuning. Moreover, we use the so-called position encoding, adjacent weight tying, and temporal encoding with 10% random noise. Stochastic gradient descent is used for training and the learning rate η is initially assigned a value of 0.005 with exponential decay applied every 25 epochs by $\eta/2$ until 100 epochs are reached. Linear start is used in all our experiments as proposed by Sukhbaatar et al. (2015). With linear start, the softmax in each memory layer is removed and re-inserted after 20 epochs. Batch size is set to 32 and gradients with an ℓ_2 norm larger than 40 are divided by a scalar to have norm 40. All weights are initialized randomly from a Gaus-

sian distribution with zero mean and $\sigma = 0.1$ except for the transform gate bias b_T^k which we empirically set the mean to 0.5. Only the most recent 50 sentences are fed into the model as the memory and the number of memory hops is 3. In all our experiments, we use the embedding size $d = 20$. Note that we re-use the same hyperparameter configuration as in (Sukhbaatar et al., 2015) and no grid search is performed.

As a large variance in the performance of the model can be observed on some tasks, we follow (Sukhbaatar et al., 2015) and repeat each training 100 times with different random initializations and select the best system based on the validation performance. On the 10k dataset, we repeat each training 30 times due to time constraints. Concerning the model implementation, while there are minor differences between the results of our implementation of MemN2N and those reported in (Sukhbaatar et al., 2015), the overall performance is equally competitive and, in some cases, better. It should be noted that v1.1 of the dataset was used whereas in this work, we employ the latest v1.2. It is therefore deemed necessary that we present the performance results of our implementation of MemN2N on the v1.2 dataset. To facilitate fair comparison, we select our implementation of MemN2N as the baseline as we believe that it is indicative of the true performance of MemN2N on v1.2 of the dataset.

Task	1k				10k			
	MemN2N	Our	GMemN2N		MemN2N	Our	GMemN2N	
		MemN2N	+global	+hop		MemN2N	+global	+hop
1: 1 supporting fact	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
2: 2 supporting facts	91.7	89.9	88.7	91.9	99.7	99.7	100.0	100.0
3: 3 supporting facts	59.7	58.5	53.2	61.2	90.7	89.1	94.7	95.5
4: 2 argument relations	97.2	99.0	99.3	99.6	100.0	100.0	100.0	100.0
5: 3 argument relations	86.9	86.6	98.1	99.0	99.4	99.4	99.9	99.8
6: yes/no questions	92.4	92.1	92.0	91.6	100.0	100.0	96.7	100.0
7: counting	82.7	83.3	83.8	82.2	96.3	96.8	96.7	98.2
8: lists/sets	90.0	89.0	87.8	87.5	99.2	98.1	99.9	99.7
9: simple negation	86.8	90.3	88.2	89.3	99.2	99.1	100.0	100.0
10: indefinite knowledge	84.9	84.6	80.1	83.5	97.6	98.0	99.9	99.8
11: basic coreference	99.1	99.7	99.8	100.0	100.0	100.0	100.0	100.0
12: conjunction	99.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
13: compound coreference	99.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
14: time reasoning	98.3	99.6	98.5	98.8	100.0	100.0	100.0	100.0
15: basic deduction	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
16: basic induction	98.7	99.9	99.8	99.9	99.6	100.0	100.0	100.0
17: positional reasoning	49.0	48.1	60.2	58.3	59.3	62.1	68.8	72.2
18: size reasoning	88.9	89.7	91.8	90.8	93.3	93.4	92.0	91.5
19: path finding	17.2	11.3	10.3	11.5	33.5	47.2	54.8	69.0
20: agent’s motivation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Average	86.1	86.1	86.6	87.3	93.4	94.1	95.2	96.3

Table 1: Accuracy (%) on the 20 QA tasks for models using 1k and 10k training examples. MemN2N:(Sukhbaatar et al., 2015). Our MemN2N: our implementation of MemN2N. GMemN2N +global: GMemN2N with global weight tying. GMemN2N +hop: GMemN2N with hop-specific weight tying. **Bold** highlights best performance. Note that in (Sukhbaatar et al., 2015), v1.1 of the dataset was used.

4.3 Results

Performance results on the 20 bAbI QA dataset are presented in Table 1. For comparison purposes, we still present MemN2N (Sukhbaatar et al., 2015) in Table 1 but accompany it with the accuracy obtained by our implementation of the same model with the same experimental setup on v1.2 of the dataset in the column “Our MemN2N” for both the 1k and 10k versions of the dataset. In contrast, we also list the results achieved by GMemN2N with global and hop-specific weight constraints in the GMemN2N columns.

GMemN2N achieves substantial improvements on task 5 and 17. The performance of GMemN2N is greatly improved, a substantial gain of more than 10 in absolute accuracy.

Global vs. hop-specific weight tying. Compared with the global weight tying scheme on the weight matrices of the gating mechanism, applying weight constraints in a hop-specific fashion

generates a further boost in performance consistently on both the 1k and 10k datasets.

State-of-the-art performance on both the 1k and 10k dataset. The best performing GMemN2N model achieves state-of-the-art performance, an average accuracy of 87.3 on the 1k dataset and 96.3 on the 10k variant. This is a solid improvement compared to MemN2N and a step closer to the strongly supervised models described in (Weston et al., 2015). Notice that the highest average accuracy of the original MemN2N model on the 10k dataset is 95.8. However, it was attained by a model with layer-wise weight tying, not adjacent weight tying as adopted in this work, and, more importantly, a much larger embedding size $d = 100$ (therefore not shown in Table 1). In comparison, it is worth noting that the proposed GMemN2N model, a much smaller model with embeddings of size 20, is capable of achieving better accuracy.

5 Dialog bAbI Experiments

In addition to the text understanding and reasoning tasks presented in Section 4, we further examine the effectiveness of the proposed GMemN2N model on a collection of goal-oriented dialog tasks (Bordes and Weston, 2016). First, we briefly describe the dataset. Next, we outline the training details. Finally, experimental results are presented with analyses.

5.1 Dataset and Data Preprocessing

In this work, we adopt the goal-oriented dialog dataset developed by Bordes and Weston (2016) organized as a set of tasks. The tasks in this dataset can be divided into 6 categories with each group focusing on a specific objective: 1. issuing API calls, 2. updating API calls, 3. displaying options, 4. providing extra-information, 5. conducting full dialogs (the aggregation of the first 4 tasks), 6. Dialog State Tracking Challenge 2 corpus (DSTC-2). The first 5 tasks are synthetically generated based on a knowledge base consisting of facts which define all the restaurants and their associated properties (7 types, such as location and price range). The generated texts are in the form of conversation between a user and a bot, each of which is designed with a clear yet different objective (all involved in a restaurant reservation scenario). This dataset essentially tests the capacity of end-to-end dialog systems to conduct dialog with various goals. Each dialog starts with a user request with subsequent alternating user-bot utterances and it is the duty of a model to understand the intention of the user and respond accordingly. In order to test the capability of a system to cope with entities not appearing in the training set, a different set of test sets, named out-of-vocabulary (OOV) test sets, are constructed separately. In addition, a supplementary dataset, task 6, is provided with real human-bot conversations, also in the restaurant domain, which is derived from the second Dialog State Tracking Challenge (Henderson et al., 2014). It is important to notice that the answers in this dataset may no longer be a single word but can be comprised of multiple ones.

5.2 Training Details

At a certain given time t , a memory-based model takes the sequence of utterances $c_1^u, c_1^r, c_2^u, c_2^r, \dots, c_{t-1}^u, c_{t-1}^r$ (alternating between the user c_i^u and the system response c_i^r) as

the stories and c_t^u as the question. The goal of the model is to predict the response c_t^r .

As answers may be composed of multiple words, following (Bordes and Weston, 2016), we replace the final prediction step in Equation (4) with:

$$\hat{a} = \text{softmax}(\mathbf{u}^\top \mathbf{W}' \Phi(\mathbf{y}_1), \dots, \mathbf{u}^\top \mathbf{W}' \Phi(\mathbf{y}_{|C|}))$$

where $\mathbf{W}' \in \mathbb{R}^{d \times |V|}$ is the weight parameter matrix for the model to learn, $\mathbf{u} = \mathbf{o}^K + \mathbf{u}^K$ (K is the total number of hops), \mathbf{y}_i is the i^{th} response in the candidate set C such that $\mathbf{y}_i \in C$, $|C|$ the size of the candidate set, and $\Phi(\cdot)$ a function which maps the input text into a bag of dimension $|V|$.

As in (Bordes and Weston, 2016), we extend Φ by several key additional features. First, two features marking the identity of the speaker of a particular utterance (user or model) are added to each of the memory slots. Second, we expand the feature representation function Φ of candidate responses with 7 additional features, each, focusing on one of the 7 properties associated with any restaurants, indicating whether there are any exact matches between words occurring in the candidate and those in the question or memory. These 7 features are referred to as the *match* features.

Apart from the modifications described above, we carry out the experiments using the same experimental setup described in Section 4.2. We also constrain ourselves to the hop-specific weight tying scheme in all our experiments since GMemN2N benefits more from it than global weight tying as shown in Section 4.3. As in (Sukhbaatar et al., 2015), since the memory-based models are sensitive to parameter initialization, we repeat each training 10 times and choose the best system based on the performance on the validation set.

5.3 Results

Performance results on the Dialog bAbI dataset are shown in Table 2, measured using both per-response accuracy and per-dialog accuracy (given in parentheses). While per-response accuracy calculates the percentage of correct responses, per-dialog accuracy, where a dialog is considered to be correct if and only if every response within it is correct, counts the percentage of correct dialogs. Task 1-5 are presented in the upper half of the table while the same tasks in the OOV setting are in the lower half with the dialog state tracking task as task 6 at the bottom. We

Task	MemN2N	GMemN2N	MemN2N +match	GMemN2N +match
T1: Issuing API calls	99.9 (99.6)	100.0 (100.0)	100.0 (100.0)	100.0 (100.0)
T2: Updating API calls	100.0 (100.0)	100.0 (100.0)	98.3 (83.9)	100.0 (100.0)
T3: Displaying options	74.9 (2.0)	74.9 (0.0)	74.9 (0.0)	74.9 (0.0)
T4: Providing information	59.5 (3.0)	57.2 (0.0)	100.0 (100.0)	100.0 (100.0)
T5: Full dialogs	96.1 (49.4)	96.3 (52.5)	93.4 (19.7)	98.0 (72.5)
Average	86.1 (50.8)	85.7 (50.5)	93.3 (60.7)	94.6 (74.5)
T1 (OOV): Issuing API calls	72.3 (0.0)	82.4 (0.0)	96.5 (82.7)	100.0 (100.0)
T2 (OOV): Updating API calls	78.9 (0.0)	78.9 (0.0)	94.5 (48.4)	94.2 (47.1)
T3 (OOV): Displaying options	74.4 (0.0)	75.3 (0.0)	75.2 (0.0)	75.1 (0.0)
T4 (OOV): Providing information	57.6 (0.0)	57.0 (0.0)	100.0 (100.0)	100.0 (100.0)
T5 (OOV): Full dialogs	65.5 (0.0)	66.7 (0.0)	77.7 (0.0)	79.4 (0.0)
Average	69.7 (0.0)	72.1 (0.0)	88.8 (46.2)	89.7 (49.4)
T6: Dialog state tracking 2	41.1 (0.0)	47.4 (1.4)	41.0 (0.0)	48.7 (1.4)

Table 2: Per-response accuracy and per-dialog accuracy (in parentheses) on the `Dialog bAbI` tasks. MemN2N: (Bordes and Weston, 2016). +match indicates the use of the match features in Section 5.2.

choose (Bordes and Weston, 2016) as the baseline which achieves the current state of the art on these tasks.

GMemN2N with the match features sets a new state of the art on most of the tasks. Other than on task T2 (OOV) and T3 (OOV), GMemN2N with the match features scores the best per-response and per-dialog accuracy. Even on T2 (OOV) and T3 (OOV), the model generates rather competitive results and remains within 0.3% of the best performance. Overall, the best average per-response accuracy in both the OOV and non-OOV categories is attained by GMemN2N.

GMemN2N with the match features significantly improves per-dialog accuracy on T5. A breakthrough in per-dialog accuracy on T5 from less than 20% to over 70%.

GMemN2N succeeds in improving the performance on the more practical task T6. With or without the match features, GMemN2N achieves a substantial boost in per-response accuracy on T6. Given that T6 is derived from a dataset based on real human-bot conversations, not synthetically generated, the performance gain, although far from perfect, highlights the effectiveness of GMemN2N in practical scenarios and constitutes an encouraging starting point towards end-to-end dialog system learning.

The effectiveness of GMemN2N is more pronounced on the more challenging tasks. The

performance gains on T5, T5 (OOV) and T6, compared with the rest of the tasks, are more pronounced. Regarding the performance of MemN2N, these tasks are relatively more challenging than the rest, suggesting that the adaptive gating mechanism in GMemN2N is capable of managing complex information flow while doing little damage on easier tasks.

6 Visualization and Analysis

In addition to the quantitative results, we further look into the memory regulation mechanism learned by the GMemN2N model. Figure 2 presents the three most frequently observed patterns of the $T^k(\mathbf{u}^k)$ vectors for each of the 3 hops in a model trained on T6 of the `Dialog bAbI` dataset with an embedding dimension of 20. Each row corresponds to the gate values at a specific hop whereas each column represents a given embedding dimension. The pattern on the top indicates that the model tends to only access memory in the first and third hop. In contrast, the middle and bottom patterns only focus on the memory in either the first or last hop respectively. Figure 3 is a t-SNE projection (Maaten and Hinton, 2008) of the flattened $[T^1(\mathbf{u}^1); T^2(\mathbf{u}^2); T^3(\mathbf{u}^3)]$ vectors obtained on the test set of the same dialog task with points corresponding to the correct and incorrect responses in red and blue respectively. Despite the relative uniform distribution of the wrong answer points, the correct ones tend to form clusters that suggest the frequently observed behavior of a successful

Story	Support	MemN2N			GMemN2N		
		Hop 1	Hop 2	Hop 3	Hop 1	Hop 2	Hop 3
Fred took the football there.	yes	0.05	0.10	0.07	0.06	0.00	0.00
Fred journeyed to the hallway.		0.45	0.09	0.01	0.00	0.00	0.00
Fred passed the football to Mary.		0.10	0.64	0.93	0.29	1.00	1.00
Mary dropped the football.		0.40	0.17	0.00	0.64	0.00	0.00
Avg. transform gate cell values, $\sum_i T^k(\mathbf{u}^k)_i/d$		N/A	N/A	N/A	0.22	0.23	0.45
Question: Who gave the football? Answer: Fred, MemN2N: Mary , GMemN2N: Fred							

Table 3: MemN2N vs. GMemN2N- bAbI dataset - Task 5 - 3 argument relations

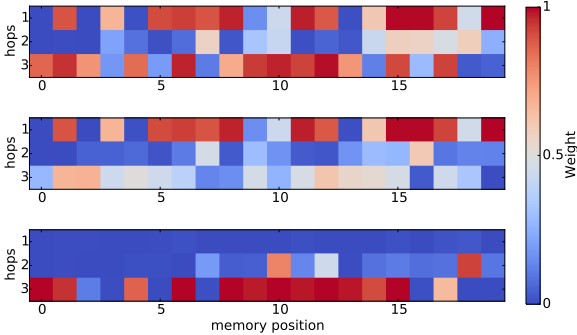


Figure 2: 3 most frequently observed gate value $T^k(\mathbf{u}^k)$ patterns on T6 of the Dialog bAbI dataset

inference. Lastly, Table 3 shows the comparison of the attention shifting process between MemN2N and GMemN2N on a story on bAbI task 5 (3 argument relations). Not only does GMemN2N manage to focus more accurately on the supporting fact than MemN2N, it has also learned to rely less in this case on hop 1 and 2 by assigning smaller transform gate values. In contrast, MemN2N carries false and misleading information (caused by the distracting attention mechanism) accumulated from the previous hops, which eventually led to the wrong prediction of the answer.

7 Related Reading Tasks

Apart from the datasets adopted in our experiments, the CNN/Daily Mail (Hermann et al., 2015) has been used for the task of machine reading formalized as a problem of text extraction from a source conditioned on a given question. However, as pointed out in (Chen et al., 2016), this dataset not only is noisy but also requires little reasoning and inference, which is evidenced by a manual analysis of a randomly selected subset of the questions, showing that only 2% of the examples call for multi-sentence inference. Richard-

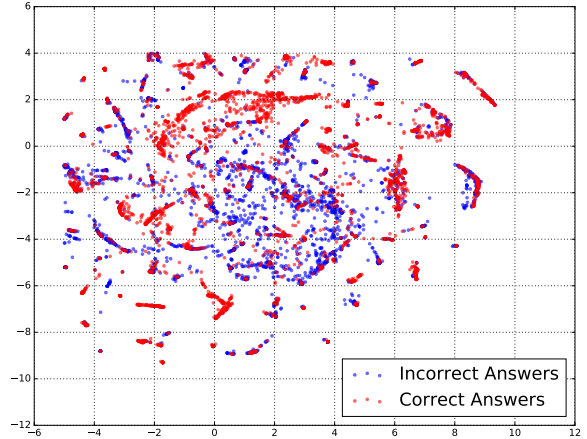


Figure 3: t-SNE scatter plot of the flattened gate values

son et al. (2013) constructed an open-domain reading comprehension task, named MCTest. Although this corpus demands various degrees of reasoning capabilities from multiple sentences, its rather limited size (660 paragraphs, each associated with 4 questions) renders training statistical models infeasible (Chen et al., 2016). Children’s Book Test (CBT) (Hill et al., 2015) was designed to measure the ability of models to exploit a wide range of linguistic context. Despite the claim in (Sukhbaatar et al., 2015) that increasing the number of hops is crucial for the performance improvements on some tasks, which can be seen as enabling MemN2N to accommodate more supporting facts, making such performance boost particularly more pronounced on those tasks requiring complex reasoning, Hill et al. (2015) admittedly reported little improvement in performance by stacking more hops and chose a single-hop MemN2N. This suggests that the necessity of multi-sentence based reasoning in this dataset is not mandatory. In the future, we plan to investigate into larger dialog datasets such as (Lowe et al., 2015).

8 Conclusion and Future Work

In this paper, we have proposed and developed what is, as far as our knowledge goes, the first attempt at incorporating an iterative memory access control to an end-to-end trainable memory-enhanced neural network architecture. We showed the added value of our proposition on a set of, natural language based, state-of-the-art reasoning tasks. Then, we offered a first interpretation of the resulting capability by analyzing the attention shifting mechanism and connection short-cutting behavior of the proposed model. In future work, we will investigate the use of such mechanism in the field of language modeling and more generally on the paradigm of sequential prediction and predictive learning. Furthermore, we plan to look into the impact of this method on the recently introduced Key-Value Memory Networks (Miller et al., 2016) on larger and semi-structured corpus.

References

- Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.
- Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.
- Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 2358–2367, Berlin, Germany.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 249–256, Sardinia, Italy.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV 2015)*, pages 1026–1034, Santiago, Chile.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, Las Vegas, USA.
- Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2014)*, pages 263–272, Philadelphia, USA.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2015)*, pages 1684–1692, Barcelona, Spain.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children’s books with explicit memory representations. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, New York, USA.
- Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. 1998. Efficient backprop. *Neural Networks: Tricks of the Trade*, pages 9–50.
- Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2015)*, Prague, Czech Republic.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, Austin, USA.
- Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 193–203, Seattle, USA.
- Brian D. Ripley. 2007. *Pattern recognition and neural networks*. Cambridge university press.
- Andrew M Saxe, James L McClelland, and Surya Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*, Banff, Canada.

- Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015a. Highway networks. *arXiv preprint arXiv:1505.00387*.
- Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015b. Training very deep networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2015)*, pages 2377–2385, Montréal, Canada.
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2015)*, pages 2440–2448, Montréal, Canada.
- Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, USA.
- Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico.
- Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, pages 2397–2406, New York, USA.

Neural Tree Indexers for Text Understanding

Tsendsuren Munkhdalai and Hong Yu

University of Massachusetts, MA, USA

{tsendsuren.munkhdalai, hong.yu}@umassmed.edu

Abstract

Recurrent neural networks (RNNs) process input text sequentially and model the conditional transition between word tokens. In contrast, the advantages of recursive networks include that they explicitly model the compositionality and the recursive structure of natural language. However, the current recursive architecture is limited by its dependence on syntactic tree. In this paper, we introduce a robust syntactic parsing-independent tree structured model, Neural Tree Indexers (NTI) that provides a middle ground between the sequential RNNs and the syntactic tree-based recursive models. NTI constructs a *full n-ary tree* by processing the input text with its node function in a bottom-up fashion. Attention mechanism can then be applied to both structure and node function. We implemented and evaluated a binary-tree model of NTI, showing the model achieved the state-of-the-art performance on three different NLP tasks: natural language inference, answer sentence selection, and sentence classification, outperforming state-of-the-art recurrent and recursive neural networks¹.

1 Introduction

Recurrent neural networks (RNNs) have been successful for modeling sequence data (Elman, 1990). RNNs equipped with gated hidden units and internal short-term memories, such as long short-term memories (LSTM) (Hochreiter and Schmidhuber, 1997) have achieved a notable success in

¹Code for the experiments and NTI is available at <https://bitbucket.org/tsendeemts/nti>

several NLP tasks including named entity recognition (Lample et al., 2016), constituency parsing (Vinyals et al., 2015), textual entailment recognition (Rocktäschel et al., 2016), question answering (Hermann et al., 2015), and machine translation (Bahdanau et al., 2015). However, most LSTM models explored so far are sequential. It encodes text sequentially from left to right or vice versa and do not naturally support compositionality of language. Sequential LSTM models seem to learn syntactic structure from the natural language however their generalization on unseen text is relatively poor comparing with models that exploit syntactic tree structure (Bowman et al., 2015b).

Unlike sequential models, recursive neural networks compose word phrases over syntactic tree structure and have shown improved performance in sentiment analysis (Socher et al., 2013). However its dependence on a syntactic tree architecture limits practical NLP applications. In this study, we introduce Neural Tree Indexers (NTI), a class of tree structured models for NLP tasks. NTI takes a sequence of tokens and produces its representation by constructing a *full n-ary tree* in a bottom-up fashion. Each node in NTI is associated with one of the node transformation functions: leaf node mapping and non-leaf node composition functions. Unlike previous recursive models, the tree structure for NTI is relaxed, i.e., NTI does not require the input sequences to be parsed syntactically; and therefore it is flexible and can be directly applied to a wide range of NLP tasks beyond sentence modeling.

Furthermore, we propose different variants of node composition function and attention over tree for our NTI models. When a sequential leaf node transformer such as LSTM is chosen, the NTI network forms a sequence-tree hybrid model taking advantage of both conditional and compositional powers of sequential and recursive models. Figure

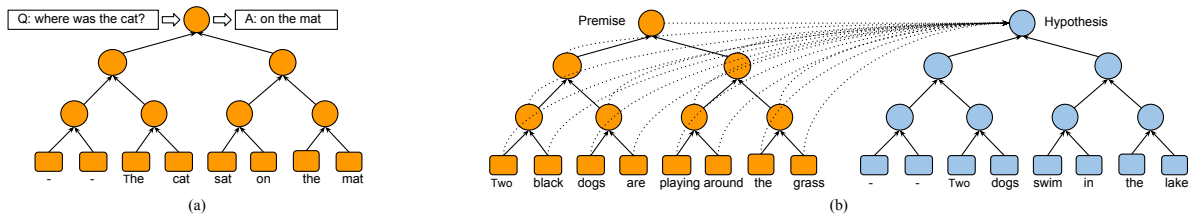


Figure 1: A binary tree form of Neural Tree Indexers (NTI) in the context of question answering and natural language inference. We insert empty tokens (denoted by $-$) to the input text to form a full binary tree. (a) NTI produces answer representation at the root node. This representation along with the question is used to find the answer. (b) NTI learns representations for the premise and hypothesis sentences and then attentively combines them for classification. Dotted lines indicate attention over premise-indexed tree.

1 shows a binary-tree model of NTI. Although the model does not follow the syntactic tree structure, we empirically show that it achieved the state-of-the-art performance on three different NLP applications: natural language inference, answer sentence selection, and sentence classification.

2 Related Work

2.1 Recurrent Neural Networks and Attention Mechanism

RNNs model input text sequentially by taking a single token at each time step and producing a corresponding hidden state. The hidden state is then passed along through the next time step to provide historical sequence information. Although a great success in a variety of tasks, RNNs have limitations (Bengio et al., 1994; Hochreiter, 1998). Among them, it is not efficient at memorizing long or distant sequence (Sutskever et al., 2014). This is frequently called as information flow bottleneck. Approaches have therefore been developed to overcome the limitations. For example, to mitigate the information flow bottleneck, Bahdanau et al. (2015) extended RNNs with a soft attention mechanism in the context of neural machine translation, leading to improved the results in translating longer sentences.

RNNs are linear chain-structured; this limits its potential for natural language which can be represented by complex structures including syntactic structure. In this study, we propose models to mitigate this limitation.

2.2 Recursive Neural Networks

Unlike RNNs, recursive neural networks explicitly model the compositionality and the recursive structure of natural language over tree. The

tree structure can be predefined by a syntactic parser (Socher et al., 2013). Each non-leaf tree node is associated with a node composition function which combines its children nodes and produces its own representation. The model is then trained by back-propagating error through structures (Goller and Kuchler, 1996).

The node composition function can be varied. A single layer network with \tanh non-linearity was adopted in recursive auto-associate memories (Pollack, 1990) and recursive autoencoders (Socher et al., 2011). Socher et al. (2012) extended this network with an additional matrix representation for each node to augment the expressive power of the model. Tensor networks have also been used as composition function for sentence-level sentiment analysis task (Socher et al., 2013). Recently, Zhu et al. (2015) introduced S-LSTM which extends LSTM units to compose tree nodes in a recursive fashion.

In this paper, we introduce a novel attentive node composition function that is based on S-LSTM. Our NTI model does not rely on either a parser output or a fine-grained supervision of non-leaf nodes, both required in previous work. In NTI, the supervision from the target labels is provided at the root node. As such, our NTI model is robust and applicable to a wide range of NLP tasks. We introduce attention over tree in NTI to overcome the vanishing/explode gradients challenges as shown in RNNs.

3 Methods

Our training set consists of N examples $\{X^i, Y^i\}_{i=1}^N$, where the input X^i is a sequence of word tokens $w_1^i, w_2^i, \dots, w_{T_i}^i$ and the output Y^i can be either a single target or a sequence. Each

input word token w_t is represented by its word embedding $x_t \in R^k$.

NTI is a *full n-ary tree* (and the sub-trees can be overlapped). It has two types of transformation function: non-leaf node function $f^{node}(h^1, \dots, h^c)$ and leaf node function $f^{leaf}(x_t)$. $f^{leaf}(x_t)$ computes a (possibly non-linear) transformation of the input word embedding x_t . $f^{node}(h^1, \dots, h^c)$ is a function of its child nodes representation h^1, \dots, h^c , where c is the total number of child nodes of this non-leaf node.

NTI can be implemented with different tree structures. In this study we implemented and evaluated a binary tree form of NTI: a non-leaf node can take in only two direct child nodes (i.e., $c = 2$). Therefore, the function $f^{node}(h^l, h^r)$ composes its left child node h^l and right child node h^r . Figure 1 illustrates our NTI model that is applied to question answering (a) and natural language inference tasks (b). Note that the node and leaf node functions are neural networks and are the only training parameters in NTI.

We explored two different approaches to compose node representations: an extended LSTM and attentive node composition functions, to be described below.

3.1 Non-Leaf Node Composition Functions

We define two different methods for non-leaf node function $f^{node}(h^l, h^r)$.

LSTM-based Non-leaf Node Function (S-LSTM): We initiate $f^{node}(h^l, h^r)$ with LSTM. For non-leaf node, we adopt S-LSTM Zhu et al. (2015), an extension of LSTM to tree structures, to learn a node representation by its children nodes. Let h_t^l, h_t^r, c_t^l and c_t^r be vector representations and cell states for the left and right children. An S-LSTM computes a parent node representation h_{t+1}^p and a node cell state c_{t+1}^p as

$$i_{t+1} = \sigma(W_1^s h_t^l + W_2^s h_t^r + W_3^s c_t^l + W_4^s c_t^r) \quad (1)$$

$$f_{t+1}^l = \sigma(W_5^s h_t^l + W_6^s h_t^r + W_7^s c_t^l + W_8^s c_t^r) \quad (2)$$

$$f_{t+1}^r = \sigma(W_9^s h_t^l + W_{10}^s h_t^r + W_{11}^s c_t^l + W_{12}^s c_t^r) \quad (3)$$

$$c_{t+1}^p = f_{t+1}^l \odot c_t^l + f_{t+1}^r \odot c_t^r + i_{t+1} \odot \tanh(W_{13}^s h_t^l + W_{14}^s h_t^r) \quad (4)$$

$$o_{t+1} = \sigma(W_{15}^s h_t^l + W_{16}^s h_t^r + W_{18}^s c_{t+1}^p) \quad (5)$$

$$h_{t+1}^p = o_{t+1} \odot \tanh(c_{t+1}^p) \quad (6)$$

where $W_1^s, \dots, W_{18}^s \in R^{k \times k}$ and biases (for brevity we eliminated the bias terms) are the training parameters. σ and \odot denote the element-wise *sigmoid* function and the element-wise vector multiplication. Extension of S-LSTM non-leaf node function to compose more children is straightforward. However, the number of parameters increases quadratically in S-LSTM as we add more child nodes.

Attentive Non-leaf Node Function (ANF): Some NLP applications (e.g., QA and machine translation) would benefit from a dynamic query dependent composition function. We introduce ANF as a new non-leaf node function. Unlike S-LSTM, ANF composes the child nodes attentively in respect to another relevant input vector $q \in R^k$. The input vector q can be a learnable representation from a sequence representation. Given a matrix $S^{ANF} \in R^{k \times 2}$ resulted by concatenating the child node representations h_t^l, h_t^r and the third input vector q , ANF is defined as

$$m = f^{score}(S^{ANF}, q) \quad (7)$$

$$\alpha = softmax(m) \quad (8)$$

$$z = S^{ANF} \alpha^\top \quad (9)$$

$$h_{t+1}^p = ReLU(W_1^{ANF} z) \quad (10)$$

where $W_1^{ANF} \in R^{k \times k}$ is a learnable matrix, $m \in R^2$ the attention score and $\alpha \in R^2$ the attention weight vector for each child. f^{score} is an attention scoring function, which can be implemented as a multi-layer perceptron (MLP)

$$m = w^\top ReLU(W_1^{score} S^{ANF} + W_2^{score} q \otimes e) \quad (11)$$

or a matrix-vector product $m = q^\top S^{ANF}$. The matrices W_1^{score} and $W_2^{score} \in R^{k \times k}$ and the vector $w \in R^k$ are training parameters. $e \in R^2$ is a vector of ones and \otimes the outer product. We use *ReLU* function for non-linear transformation.

3.2 Attention Over Tree

Comparing with sequential LSTM models, NTI has less recurrence, which is defined by the tree depth, $\log(n)$ for binary tree where n is the length of the input sequence. However, NTI still needs to compress all the input information into a single representation vector of the root. This imposes practical difficulties when processing long sequences. We address this issue with attention

Model	d	$ \theta _M$	Train	Test
Classifier with handcrafted features (Bowman et al., 2015a)	-	-	99.7	78.2
LSTMs encoders (Bowman et al., 2015a)	300	3.0M	83.9	80.6
Dependency Tree CNN encoders (Mou et al., 2016)	300	3.5M	83.3	82.1
NTI-SLSTM (Ours)	300	3.3M	83.9	82.4
SPINN-PI encoders (Bowman et al., 2016)	300	3.7M	89.2	83.2
NTI-SLSTM-LSTM (Ours)	300	4.0M	82.5	83.4
LSTMs attention (Rocktäschel et al., 2016)	100	242K	85.4	82.3
LSTMs word-by-word attention (Rocktäschel et al., 2016)	100	250K	85.3	83.5
NTI-SLSTM node-by-node global attention (Ours)	300	3.5M	85.0	84.2
NTI-SLSTM node-by-node tree attention (Ours)	300	3.5M	86.0	84.3
NTI-SLSTM-LSTM node-by-node tree attention (Ours)	300	4.2M	88.1	85.7
NTI-SLSTM-LSTM node-by-node global attention (Ours)	300	4.2M	87.6	85.9
mLSTM word-by-word attention (Wang and Jiang, 2016)	300	1.9M	92.0	86.1
LSTMN with deep attention fusion (Cheng et al., 2016)	450	3.4M	88.5	86.3
Tree matching NTI-SLSTM-LSTM tree attention (Ours)	300	3.2M	87.3	86.4
Decomposable Attention Model (Parikh et al., 2016)	200	580K	90.5	86.8
Tree matching NTI-SLSTM-LSTM global attention (Ours)	300	3.2M	87.6	87.1
Full tree matching NTI-SLSTM-LSTM global attention (Ours)	300	3.2M	88.5	87.3

Table 1: Training and test accuracy on natural language inference task. d is the word embedding size and $|\theta|_M$ the number of model parameters.

mechanism over tree. In addition, the attention mechanism can be used for matching trees (described in Section 4 as Tree matching NTI) that carry different sequence information. We first define a global attention and then introduce a tree attention which considers the parent-child dependency for calculation of the attention weights.

Global Attention: An attention neural network for the global attention takes all node representations as input and produces an attentively blended vector for the whole tree. This neural net is similar to ANF. Particularly, given a matrix $S^{GA} \in R^{k \times 2n-1}$ resulted by concatenating the node representations h_1, \dots, h_{2n-1} and the relevant input representation q , the global attention is defined as

$$m = f^{score}(S^{GA}, q) \quad (12)$$

$$\alpha = softmax(m) \quad (13)$$

$$z = S^{GA} \alpha^\top \quad (14)$$

$$h^{tree} = ReLU(W_1^{GA} z + W_2^{GA} q) \quad (15)$$

where W_1^{GA} and $W_2^{GA} \in R^{k \times k}$ are training parameters and $\alpha \in R^{2n-1}$ the attention weight vector for each node. This attention mechanism is robust as it globally normalizes the attention score m with $softmax$ to obtain the weights α . However, it does not consider the tree structure when producing the final representation h^{tree} .

Tree Attention: We modify the global attention network to the tree attention mechanism. The resulting tree attention network performs almost the same computation as ANF for each node. It

compares the parent and children nodes to produce a new representation assuming that all node representations are constructed. Given a matrix $S^{TA} \in R^{k \times 3}$ resulted by concatenating the parent node representation h_t^p , the left child h_t^l and the right child h_t^r and the relevant input representation q , every non-leaf node h_t^p simply updates its own representation by using the following equation in a bottom-up manner.

$$m = f^{score}(S^{TA}, q) \quad (16)$$

$$\alpha = softmax(m) \quad (17)$$

$$z = S^{TA} \alpha^\top \quad (18)$$

$$h_t^p = ReLU(W_1^{TA} z) \quad (19)$$

and this equation is similarity to the global attention. However, now each non-leaf node attentively collects its own and children representations and passes towards the root which finally constructs the attentively blended tree representation. Note that unlike the global attention, the tree attention locally normalizes the attention scores with $softmax$.

4 Experiments

We describe in this section experiments on three different NLP tasks, natural language inference, question answering and sentence classification to demonstrate the flexibility and the effectiveness of NTI in the different settings.

We trained NTI using Adam (Kingma and Ba, 2014) with hyperparameters selected on development set. The pre-trained 300-D Glove 840B vectors (Pennington et al., 2014) were obtained for the word embeddings². The word embeddings are fixed during training. The embeddings for out-of-vocabulary words were set to zero vector. We pad the input sequence to form a *full binary tree*. A padding vector was inserted when padding. We analyzed the effects of the padding size and found out that it has no influence on the performance (see Appendix 5.3). The size of hidden units of the NTI modules were set to 300. The models were regularized by using dropouts and an l_2 weight decay.³

4.1 Natural Language Inference

We conducted experiments on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015a), which consists of 549,367/9,842/9,824 premise-hypothesis pairs for train/dev/test sets and target label indicating their relation. Unless otherwise noted, we follow the setting in the previous work (Mou et al., 2016; Bowman et al., 2016) and use an MLP for classification which takes in NTI outputs and computes the concatenation $[h_{2n-1}^p; h_{2n-1}^h]$, absolute difference $h_{2n-1}^p - h_{2n-1}^h$ and elementwise product $h_{2n-1}^p \cdot h_{2n-1}^h$ of the two sentence representations. The MLP has also an input layer with 1024 units with *ReLU* activation and a *softmax* output layer. We explored nine different task-oriented NTI models with varying complexity, to be described below. For each model, we set the batch size to 32. The initial learning, the regularization strength and the number of epoch to be trained are varied for each model.

NTI-SLSTM: this model does not rely on f^{leaf} transformer but uses the S-LSTM units for the non-leaf node function. We set the initial learning rate to 1e-3 and l_2 regularizer strength to 3e-5, and train the model for 90 epochs. The neural net was regularized by 10% input dropouts and the 20% output dropouts.

NTI-SLSTM-LSTM: we use LSTM for the leaf node function f^{leaf} . Concretely, the LSTM output vectors are given to NTI-SLSTM and the memory cells of the lowest level S-LSTM were initialized with the LSTM memory states. The hyper-parameters are the same as the previous

model.

NTI-SLSTM node-by-node global attention: This model learns inter-sentence relation with the global attention over premise-indexed tree, which is similar to word-by-word attention model of Rocktäschel et al. (2016) in that it attends over the premise tree nodes at every time step of hypothesis encoding. We tie the weight parameters of the two NTI-SLSTMs for premise and hypothesis and no f^{leaf} transformer used. We set the initial learning rate to 3e-4 and l_2 regularizer strength to 1e-5, and train the model for 40 epochs. The neural net was regularized by 15% input dropouts and the 15% output dropouts.

NTI-SLSTM node-by-node tree attention: this is a variation of the previous model with the tree attention. The hyper-parameters are the same as the previous model.

NTI-SLSTM-LSTM node-by-node global attention: in this model we include LSTM as the leaf node function f^{leaf} . Here we initialize the memory cell of S-LSTM with LSTM memory and hidden/memory state of hypothesis LSTM with premise LSTM (the later follows the work of (Rocktäschel et al., 2016)). We set the initial learning rate to 3e-4 and l_2 regularizer strength to 1e-5, and train the model for 10 epochs. The neural net was regularized by 10% input dropouts and the 15% output dropouts.

NTI-SLSTM-LSTM node-by-node tree attention: this is a variation of the previous model with the tree attention. The hyper-parameters are the same as the previous model.

Tree matching NTI-SLSTM-LSTM global attention: this model first constructs the premise and hypothesis trees simultaneously with the NTI-SLSTM-LSTM model and then computes their matching vector by using the global attention and an additional LSTM. The attention vectors are produced at each hypothesis tree node and then are given to the LSTM model sequentially. The LSTM model compress the attention vectors and outputs a single matching vector, which is passed to an MLP for classification. The MLP for this tree matching setting has an input layer with 1024 units with *ReLU* activation and a *softmax* output layer.

Unlike Wang and Jiang (2016)’s matching LSTM model which is specific to matching sequences, we use the standard LSTM units and match trees. We set the initial learning rate to 3e-

²<http://nlp.stanford.edu/projects/glove/>

³More detail on hyper-parameters can be found in code.

4 and l_2 regularizer strength to $3e-5$, and train the model for 20 epochs. The neural net was regularized by 20% input dropouts and the 20% output dropouts.

Tree matching NTI-SLSTM-LSTM tree attention: we replace the global attention with the tree attention. The hyper-parameters are the same as the previous model.

Full tree matching NTI-SLSTM-LSTM global attention: this model produces two sets of the attention vectors, one by attending over the premise tree regarding each hypothesis tree node and another by attending over the hypothesis tree regarding each premise tree node. Each set of the attention vectors is given to a LSTM model to achieve full tree matching. The last hidden states of the two LSTM models (i.e. one for each attention vector set) are concatenated for classification. The training weights are shared among the LSTM models. The hyper-parameters are the same as the previous model.⁴

Table 1 shows the results of our models. For comparison, we include the results from the published state-of-the-art systems. While most of the sentence encoder models rely solely on word embeddings, the dependency tree CNN and the SPINN-PI models make use of sentence parser output; which present strong baseline systems. The last set of methods designs inter-sentence relation with soft attention (Bahdanau et al., 2015). Our best score on this task is 87.3% accuracy obtained with the full tree matching NTI model. The previous best performing model on the task performs phrase matching by using the attention mechanism.

Our results show that NTI-SLSTM improved the performance of the sequential LSTM encoder by approximately 2%. Not surprisingly, using LSTM as leaf node function helps in learning better representations. Our NTI-SLSTM-LSTM is a hybrid model which encodes a sequence sequentially through its leaf node function and then hierarchically composes the output representations. The node-by-node attention models improve the performance, indicating that modeling inter-sentence interaction is an important element in NLI. Aggregating matching vector between trees or sequences with a separate LSTM model is effective. The global attention seems to

⁴Computational constraint prevented us from experimenting the tree attention variant of this model

Model	MAP	MRR
Classifier with features (2013)	0.5993	0.6068
Paragraph Vector (2014)	0.5110	0.5160
Bigram-CNN (2014)	0.6190	0.6281
3-layer LSTM (2016)	0.6552	0.6747
3-layer LSTM attention (2016)	0.6639	0.6828
NASM (2016)	0.6705	0.6914
NTI (Ours)	0.6742	0.6884

Table 2: Test set performance on answer sentence selection.

Model	Bin	FG
RNTN (Socher et al., 2013)	85.4	45.7
CNN-MC (Kim, 2014)	88.1	47.4
DRNN (Irsoy and Cardie, 2015)	86.6	49.8
2-layer LSTM (Tai et al., 2015)	86.3	46.0
Bi-LSTM (Tai et al., 2015)	87.5	49.1
NTI-SLSTM (Ours)	87.8	50.5
CT-LSTM (Tai et al., 2015)	88.0	51.0
DMN (Kumar et al., 2016)	88.6	52.1
NTI-SLSTM-LSTM (Ours)	89.3	53.1

Table 3: Test accuracy for sentence classification. Bin: binary, FG: fine-grained 5 classes.

be robust on this task. The tree attention were not helpful as it normalizes the attention scores locally in parent-child relationship.

4.2 Answer Sentence Selection

For this task, a model is trained to identify the correct sentences that answer a factual question, from a set of candidate sentences. We experiment on WikiQA dataset constructed from Wikipedia (Yang et al., 2015). The dataset contains 20,360/2,733/6,165 QA pairs for train/dev/test sets.

We used the same setup in the language inference task except that we replace the *softmax* layer with a *sigmoid* layer and model the following conditional probability distribution.

$$p_{\theta}(y = 1 | h_n^q, h_n^a) = \text{sigmoid}(o^{QA}) \quad (20)$$

where h_n^q and h_n^a are the question and the answer encoded vectors and o^{QA} denotes the output of the hidden layer of the MLP. For this task, we use NTI-SLSTM-LSTM to encode answer candidate sentences and NTI-ANF-LSTM to encode the question sentences. Note that NTI-ANF-LSTM is relied on ANF as the non-leaf node function. q vector for NTI-ANF-LSTM is the answer representation produced by the answer encoding NTI-SLSTM-LSTM model. We set the batch size to 4 and the initial learning rate to $1e-3$, and train the

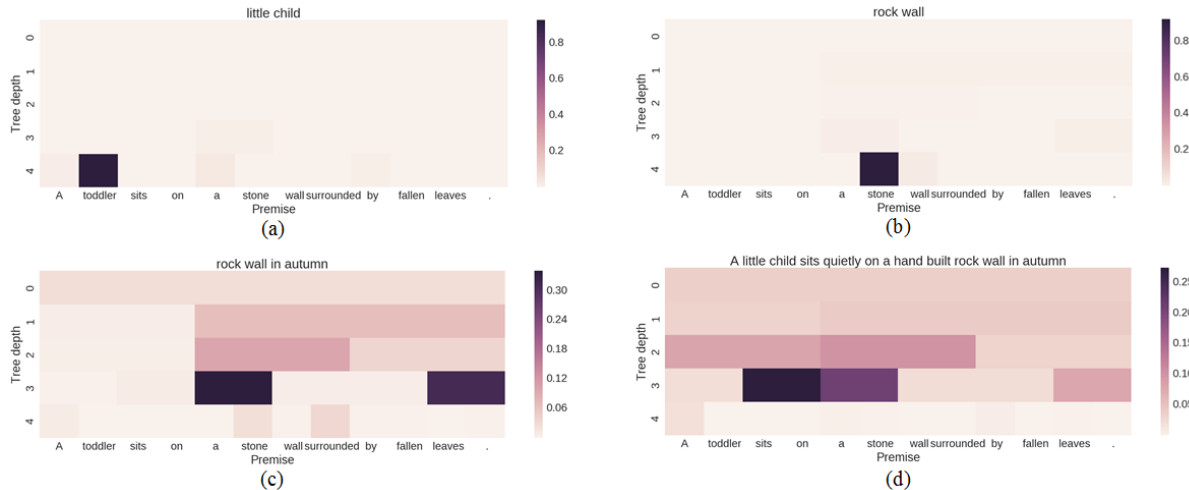


Figure 2: Node-by-node attention visualizations. The phrases shown on the top are nodes from hypothesis-indexed tree and the premise tokens are listed along the x-axis. The adjacent cells are composed in the top cell representing a binary tree and resulting a longer attention span.

a person	park for fun	Santa Claus	sad, depressed, and hatred
single person	an outdoor concert at the park	a snowmobile in a blizzard	an Obama supporter is upset
a woman	kids playing at a park outside	a Skier ski - jumping	but doesn't have any money
a young person	a mom takes a break in a park	A skier preparing a trick	crying because he didn't get cake
a guy	people play frisbee outdoors	a child is playing on christmas	trying his hardest to not fall off
a single human	takes his lunch break in the park	two men play with a snowman	is upset and crying on the ground

Table 4: Nearest-neighbor phrases based on cosine similarity between learned representations.

model for 10 epochs. We used 20% input dropouts and no l_2 weight decay. Following previous work, we adopt MAP and MRR as the evaluation metrics for this task.⁵

Table 2 presents the results of our model and the previous models for the task.⁶ The classifier with handcrafted features is a SVM model trained with a set of features. The Bigram-CNN model is a simple convolutional neural net. The Deep LSTM and LSTM attention models outperform the previous best result by a large margin, nearly 5-6%. NASM improves the result further and sets a strong baseline by combining variational auto-encoder (Kingma and Welling, 2014) with the soft attention. In NASM, they adopt a deep three-layer LSTM and introduced a latent stochastic attention mechanism over the answer sentence. Our NTI model exceeds NASM by approximately 0.4% on MAP for this task.

⁵We used *trec_eval* script to calculate the evaluation metrics

⁶Inclusion of simple word count feature improves the performance by around 0.15-0.3 across the board

4.3 Sentence Classification

Lastly, we evaluated NTI on the Stanford Sentiment Treebank (SST) (Socher et al., 2013). This dataset comes with standard train/dev/test sets and two subtasks: binary sentence classification or fine-grained classification of five classes. We trained our model on the text spans corresponding to labeled phrases in the training set and evaluated the model on the full sentences.

We use NTI-SLSTM and NTI-SLSTM-LSTM models to learn sentence representations for the task. The sentence representations were passed to a two-layer MLP for classification. We set the batch size to 64, the initial learning rate to $1e-3$ and l_2 regularizer strength to $3e-5$, and train each model for 10 epochs. The NTI-SLSTM model was regularized by 10%/20% of input/output and 20%/30% of input/output dropouts and the NTI-SLSTM-LSTM model 20% of input and 20%/30% of input/output dropouts for binary and fine-grained settings.

NTI-SLSTM-LSTM (as shown in Table 5) set the state-of-the-art results on both subtasks. Our NTI-SLSTM model performed slightly worse

A dog mouth holds a retrieved ball.	A cat nurses puppies.	A dog sells a woman a hat.
A brown and white dog holds a tennis ball in his mouth. The dog has a ball. The dogs are chasing a ball.	A golden retriever nurses some other dogs puppies. A golden retriever nurses puppies. A mother dog checking up on her baby puppy.	The dog is a labrador retriever.
A small dog runs to catch a ball. The puppy is chasing a ball.	A girl is petting her dog. The hat wearing girl is petting a cat.	A girl is petting her dog. The dog is a shitzu. A husband and wife making pizza. The dog is a chihuahua.

Table 5: Nearest-neighbor sentences based on cosine similarity between learned representations.

than its constituency tree-based counter part, CT-LSTM model. The CT-LSTM model composes phrases according to the output of a sentence parser and uses a node composition function similar to S-LSTM. After we transformed the input with the LSTM leaf node function, we achieved the best performance on this task.

5 Qualitative Analysis

5.1 Attention and Compositionality

To help analyzing the results, we output attention weights by our NTI-SLSTM node-by-node global attention model. Figure 2 shows the attention heatmaps for two sentences in the SNLI test set. It shows that our model semantically aligns single or multiword expressions (*"little child"* and *"toddler"*; *"rock wall"* and *"stone"*). In addition, our model is able to re-orient its attention over different parts of the hypothesis when the expression is more complex. For example, for (c) *"rock wall in autumn"*, NTI mostly focuses on the nodes in depth 1, 2 and 3 representing contexts related to *"a stone"*, *"leaves."* and *"a stone wall surrounded"*. Surprisingly, attention degree for the single word expression like *"stone"*, *"wall"* and *"leaves"* is lower to compare with multiword phrases. Sequence models lack this property as they have no explicit composition module to produce such mu-

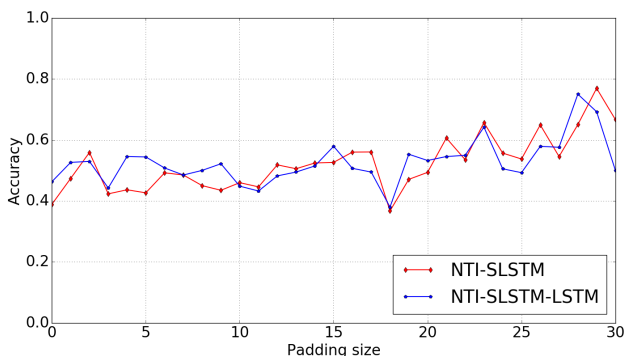


Figure 3: Fine-grained sentiment classification accuracy vs. padding size on test set of SST data.

tiword phrases.

Finally, the most interesting pattern is that the model attends over higher level (low depth) tree nodes with rich semantics when considering a (c) longer phrase or (d) full sentence. As shown in (d), the NTI model aligns the root node representing the whole hypothesis sentence to the higher level tree nodes covering larger sub-trees in the premise. It certainly ignores the lower level single word expressions and only starts to attend when the words are collectively to form rich semantics.

5.2 Learned Representations of Phrases and Sentences

Using cosine similarity between their representations produced by the NTI-SLSTM model, we show that NTI is able to capture paraphrases on SNLI test data. As shown in Table 4, NTI seems to distinguish plural from singular forms (similar phrases to *"a person"*). In addition, NTI captures non-surface knowledge. For example, the phrases similar to *"park for fun"* tend to align to the semantic content of *fun* and *park*, including *"people play frisbee outdoors"*. The NTI model was able to relate *"Santa Claus"* to *christmas* and *snow*. Interestingly, the learned representations were also able to connect implicit semantics. For example, NTI found that *"sad, depressed, and hatred"* is close to the phrases like *"an Obama supporter is upset"*. Overall the NTI model is robust to the length of the phrases being matched. Given a short phrase, NTI can retrieve longer yet semantically coherent sequences from the SNLI test set.

In Table 5, we show nearest-neighbor sentences from SNLI test set. Note that the sentences listed in the first two columns sound semantically coherent but not the ones in the last column. The query sentence *"A dog sells a woman a hat"* does not actually represent a common-sense knowledge and this sentence now seem to confuse the NTI model. As a result, the retrieved sentence are arbitrary and not coherent.

5.3 Effects of Padding Size

We introduced a special padding character in order to construct full binary tree. Does this padding character influence the performance of the NTI models? In Figure 3, we show relationship between the padding size and the accuracy on Stanford sentiment analysis data. Each sentence was padded to form a full binary tree. The x-axis represents the number of padding characters introduced. When the padding size is less (up to 10), the NTI-SLSTM-LSTM model performs better. However, this model tends to perform poorly or equally when the padding size is large. Overall we do not observe any significant performance drop for both models as the padding size increases. This suggests that NTI learns to ignore the special padding character while processing padded sentences. The same scenario was also observed while analyzing attention weights. The attention over the padded nodes was nearly zero.

6 Discussion and Conclusion

We introduced Neural Tree Indexers, a class of tree structured recursive neural network. The NTI models achieved state-of-the-art performance on different NLP tasks. Most of the NTI models form deep neural networks and we think this is one reason that NTI works well even if it lacks direct linguistic motivations followed by other syntactic-tree-structured recursive models (Socher et al., 2013).

CNN and NTI are topologically related (Kalchbrenner and Blunsom, 2013). Both NTI and CNNs are hierarchical. However, current implementation of NTI only operates on non-overlapping subtrees while CNNs can slide over the input to produce higher-level representations. NTI is flexible in selecting the node function and the attention mechanism. Like CNN, the computation in the same tree-depth can be parallelized effectively; and therefore NTI is scalable and suitable for large-scale sequence processing. Note that NTI can be seen as a generalization of LSTM. If we construct left-branching trees in a bottom-up fashion, the model acts just like sequential LSTM. Different branching factors for the underlying tree structure have yet to be explored. NTI can be extended so it learns to select and compose dynamic number of nodes for efficiency, essentially discovering intrinsic hierarchical structure in the input.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by the grant HL125089 from the National Institutes of Health (NIH). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.
- Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.
- Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015 NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches-Volume 1583*, pages 37–42.
- Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany, August. Association for Computational Linguistics.
- Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November. Association for Computational Linguistics.
- Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.
- Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS 2015*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.
- Ozan Irsoy and Claire Cardie. 2015. Modeling compositionality with multiplicative recurrent neural networks. In *ICLR 2015*.
- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR 2014*.
- Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR 2014*.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of The 33rd International Conference on Machine Learning (ICML 2016)*, pages 1378–1387, New York, NY, USA, June.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.
- Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML 2014*, volume 14, pages 1188–1196.
- Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICLR 2016*.
- Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136, Berlin, Germany, August. Association for Computational Linguistics.
- Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November. Association for Computational Linguistics.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.
- Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR 2016*.
- Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.
- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory

- networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.
- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS 2015*.
- Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, California, June. Association for Computational Linguistics.
- Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September. Association for Computational Linguistics.
- Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Deep Learning Workshop 2014*.
- Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*, pages 1604–1612.

Exploring Different Dimensions of Attention for Uncertainty Detection

Heike Adel and Hinrich Schütze

Center for Information and Language Processing (CIS)

LMU Munich, Germany

heike@cis.lmu.de

Abstract

Neural networks with attention have proven effective for many natural language processing tasks. In this paper, we develop attention mechanisms for uncertainty detection. In particular, we generalize standardly used attention mechanisms by introducing *external attention* and *sequence-preserving attention*. These novel architectures differ from standard approaches in that they use external resources to compute attention weights and preserve sequence information. We compare them to other configurations along different dimensions of attention. Our novel architectures set the new state of the art on a Wikipedia benchmark dataset and perform similar to the state-of-the-art model on a biomedical benchmark which uses a large set of linguistic features.

1 Introduction

For many natural language processing (NLP) tasks, it is essential to distinguish uncertain (non-factual) from certain (factual) information. Such tasks include information extraction, question answering, medical information retrieval, opinion detection, sentiment analysis (Karttunen and Zelenen, 2005; Vincze, 2014a; Díaz et al., 2016) and knowledge base population (KBP). In KBP, we need to distinguish, e.g., “X may be Basque” and “X was rumored to be Basque” (uncertain) from “X is Basque” (certain) to decide whether to add the fact “Basque(X)” to a knowledge base. In this paper, we use the term *uncertain information* to refer to speculation, opinion, vagueness and ambiguity. We focus our experiments on the uncertainty detection (UD) dataset from the CoNLL2010 hedge cue detection task (Farkas

et al., 2010). It consists of two medium-sized corpora from different domains (Wikipedia and biomedical) that allow us to run a large number of comparative experiments with different neural networks and exhaustively investigate different dimensions of attention.

Convolutional and recurrent neural networks (CNNs and RNNs) perform well on many NLP tasks (Collobert et al., 2011; Kalchbrenner et al., 2014; Zeng et al., 2014; Zhang and Wang, 2015). CNNs are most often used with pooling. More recently, attention mechanisms have been successfully integrated into CNNs and RNNs (Bahdanau et al., 2015; Rush et al., 2015; Hermann et al., 2015; Rocktäschel et al., 2016; Yang et al., 2016; He and Golub, 2016; Yin et al., 2016). Both pooling and attention can be thought of as *selection mechanisms* that help the network focus on the most relevant parts of a layer, either an input or a hidden layer. This is especially beneficial for long input sequences, e.g., long sentences or entire documents. We apply CNNs and RNNs to uncertainty detection and compare them to a number of baselines. We show that attention-based CNNs and RNNs are effective for uncertainty detection. On a Wikipedia benchmark, we improve the state of the art by more than 3.5 F_1 points.

Despite the success of attention in prior work, the design space of related network architectures has not been fully explored. In this paper, we develop novel ways to calculate attention weights and integrate them into neural networks. Our models are motivated by the characteristics of the uncertainty task, yet they are also a first attempt to systematize the design space of attention. In this paper, we begin with investigating three dimensions of this space: weighted vs. unweighted selection, sequence-agnostic vs. sequence-preserving selection, and internal vs. external attention.

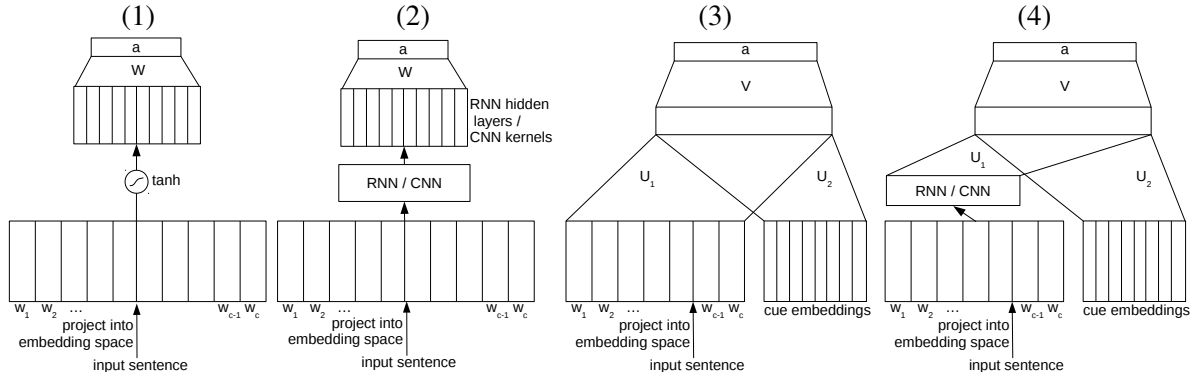


Figure 1: Internal attention on (1) input and (2) hidden representation. External attention on (3) input and (4) hidden representation. For the whole network structure, see Figure 3.

Weighted vs. Unweighted Selection. Pooling is unweighted selection: it outputs the selected values as is. In contrast, attention can be thought of as weighted selection: some input elements are highly weighted, others receive weights close to zero and are thereby effectively not selected. The advantage of weighted selection is that the model learns to decide based on the input how many values it should select. Pooling either selects all values (average pooling) or k values (k -max pooling). If there are more than k uncertainty cues in a sentence, pooling is not able to focus on all of them.

Sequence-agnostic vs. Sequence-preserving Selection. K -max pooling (Kalchbrenner et al., 2014) is sequence-preserving: it takes a long sequence as input and outputs a subsequence whose members are in the same order as in the original sequence. In contrast, attention is generally implemented as a weighted average of the input vectors. That means that all ordering information is lost and cannot be recovered by the next layer. As an alternative, we present and evaluate new sequence-preserving ways of attention. For uncertainty detection, this might help distinguishing phrases like “it is not uncertain that X is Basque” and “it is uncertain that X is not Basque”.

Internal vs. External Attention. Prior work calculates attention weights based on the input or hidden layers of the neural network. We call this internal attention. For uncertainty detection, it can be beneficial to give the model a lexicon of seed cue words or phrases. Thus, we provide the network with additional information to bear on identifying and summarizing features. This can simplify the training process by guiding the model to recognizing uncertainty cues. We call this external attention and show that it improves performance

for uncertainty detection.

Previous work on attention and pooling has only considered a small number of the possible configurations along those dimensions of attention. However, the internal/external and un/weighted distinctions can potentially impact performance because external resources add information that can be critical for good performance and because weighting increases the flexibility and expressivity of neural network models. Also, word order is often critical for meaning and is therefore an important feature in NLP. Although our models are motivated by the characteristics of uncertainty detection, they could be useful for other NLP tasks as well.

Our main contributions are as follows. (i) We extend the design space of selection mechanisms for neural networks and conduct an extensive set of experiments testing various configurations along several dimensions of that space, including novel sequence-preserving and external attention mechanisms. (ii) To our knowledge, we are the first to apply convolutional and recurrent neural networks to uncertainty detection. We demonstrate the effectiveness of the proposed attention architectures for this task and set the new state of the art on a Wikipedia benchmark dataset. (iii) We publicly release our code for future research.¹

2 Models

Convolutional Neural Networks. CNNs have been successful for many NLP tasks since convolution and pooling can detect key features independent of their position in the sentence. Moreover, they can take advantage of word embeddings and their characteristics. Both properties are also

¹<http://cistern.cis.lmu.de>

essential for uncertainty detection since we need to detect cue phrases that can occur anywhere in the sentence; and since some notion of similarity improves performance if a cue phrase in the test data did not occur in the training data, but is similar to one that did. The CNN we use in this paper has one convolutional layer, 3-max pooling (see Kalchbrenner et al. (2014)), a fully connected hidden layer and a logistic output unit.

Recurrent Neural Networks. Different types of RNNs have been applied widely to NLP tasks, including language modeling (Bengio et al., 2000; Mikolov et al., 2010), machine translation (Cho et al., 2014; Bahdanau et al., 2015), relation classification (Zhang and Wang, 2015) and entailment (Rocktäschel et al., 2016). In this paper, we apply a bi-directional gated RNN (GRU) with gradient clipping and a logistic output unit. Chung et al. (2014) showed that GRUs and LSTMs have similar performance, but GRUs are more efficient in training. The hidden layer h of the GRU is parameterized by two matrices W and U and four additional matrices W_r , U_r and W_z , U_z for the reset gate r and the update gate z (Cho et al., 2014):

$$r = \sigma(W_r x + U_r h^{t-1}) \quad (1)$$

$$z = \sigma(W_z x + U_z h^{t-1}) \quad (2)$$

$$h^t = z \odot h^{t-1} + (1 - z) \odot \tilde{h}^t \quad (3)$$

$$\tilde{h}^t = \sigma(W x + U(r \odot h^{t-1})) \quad (4)$$

t is the index for the current time step, \odot is element-wise multiplication and σ is the sigmoid.

3 Attention

3.1 Architecture of the Attention Layer

We first define an attention layer a for input x :

$$\alpha_i = \frac{\exp(f(x_i))}{\sum_j \exp(f(x_j))} \quad (5)$$

$$a_i = \alpha_i \cdot x_i \quad (6)$$

where f is a scoring function, the α_i are the attention weights and each input x_i is reweighted by its corresponding attention weight α_i .

The most basic definition of f is as a linear scoring function on the input x :

$$f(x_i) = W^T x_i \quad (7)$$

W are parameters that are learned in training.

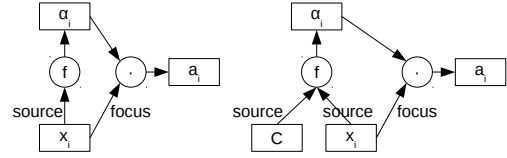


Figure 2: Schemes of focus and source: left: internal attention, right: external attention

3.2 Focus and Source of Attention

In this paper, we distinguish between focus and source of attention.

The *focus* of attention is the layer of the network that is reweighted by attention weights, corresponding to x in Eq. 6. We consider two options for the application in uncertainty detection as shown in Figure 1: (i) the focus is on the input, i.e., the matrix of word vectors ((1) and (3)) and (ii) the focus is on the convolutional layer of the CNN or the hidden layers of the RNN ((2) and (4)). For focus on the input, we apply \tanh to the word vectors (see part (1) of figure) to improve results.

The *source* of attention is the information source that is used to compute the attention weights, corresponding to the input of f in Eq. 5.

Eq. 7 formalizes the case in which focus and source are identical (both are based only on x). We call this **internal attention** (see left part of Figure 2). *An attention layer is called internal if both focus and source are based only on information internally available to the network (through input or hidden layers).*²

If we conceptualize attention in terms of source and focus, then a question that arises is whether we can make it more powerful by *increasing the scope of the source beyond the input*.

In this paper, we propose a way of expanding the source of attention by making an *external resource* C available to the scoring function f :

$$f(x_i) = f'(x_i, C) \quad (8)$$

We call this **external attention** (see right part of Figure 2). *An attention layer is called external if its source includes an external resource.*

The specific external-attention scoring function we use for uncertainty detection is parametrized by U_1 , U_2 and V and defined as follows:

$$f(x_i) = \sum_j V^T \cdot \tanh(U_1 \cdot x_i + U_2 \cdot c_j) \quad (9)$$

²Gates, e.g., the weighting of h^{t-1} in Eq. 4, can also be viewed as internal attention mechanisms.

where c_j is a vector representing a cue phrase j of the training set. We compute c_j as the average of the embeddings of the constituent words of j .

This attention layer scores an input word x_i by comparing it with each cue vector c_j and summing the results. The comparison is done using a fully connected hidden layer. Its weights U_1, U_2 and V are learned during training. When using this scoring function in Eq. 5, each α_i is an assessment of how important x_i is for uncertainty detection, taking into account our knowledge about cue phrases. Since we use embeddings to represent words and cues, uncertainty-indicating phrases that did not occur in training, but are similar to training cue phrases can also be recognized.

We use this novel attention mechanism for uncertainty detection, but it is also applicable to other tasks and domains as long as there is a set of vectors available that is analogous to our c_j vectors, i.e., vectors that model relevance of embeddings to the task at hand (for an outlook, see Section 6).

3.3 Sequence-agnostic vs. Sequence-preserving Selection

So far, we have explained the basic architecture of an attention layer: computing attention weights and reweighting the input. We now turn to the *integration of the attention layer* into the overall network architecture, i.e., how it is connected to downstream components.

The most frequently used downstream connection of the attention layer is to take the **average**:

$$a = \sum_i \alpha_i \quad (10)$$

We call this the average, not the sum, because the α_i are normalized to sum to 1 and the standard term for this is “weighted average”.

A variant is the **k-max average**:

$$a = \sum_{R(\alpha_j) \leq k} \alpha_j$$

where $R(\alpha_j)$ is the rank of α_j in the list of activation weights α_i in descending order. This type of averaging is more similar to k-max pooling and may be more robust because elements with low weights (which may just be noise) will be ignored.

Averaging destroys order information that may be needed for NLP sequence classification tasks. Therefore, we also investigate a sequence-preserving method, **k-max sequence**:

$$a = [a_j | R(\alpha_j) \leq k] \quad (11)$$

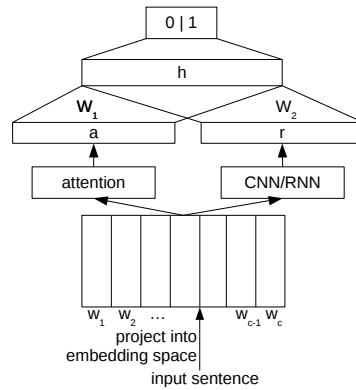


Figure 3: Network overview: combination of attention and CNN/RNN output. For details on attention, see Figure 1.

where $[a_j | P(a_j)]$ denotes the subsequence of sequence $A = [a_1, \dots, a_J]$ from which members not satisfying predicate P have been removed. Note that sequence a is in the original order of the input, i.e., not sorted by value.

K-max sequence selects a subsequence of input vectors. Our last integration method is **k-max pooling**. It ranks each dimension of the vectors individually, thus the resulting values can stem from different input positions. This is the same as standard k-max pooling in CNNs except that each vector element in a_j has been weighted (by its attention weight α_j), whereas in standard k-max pooling it is considered as is. Below, we also refer to k-max sequence as “per-pos” and to k-max pooling as “per-dim” to clearly distinguish it from k-max pooling done by the CNN.

Combination with CNN and RNN Output.

Another question is whether we combine the attention result with the result of the convolutional or recurrent layer of the network. Since k-max pooling (CNN) and recurrent hidden layers with gates (RNN) have strengths complementary to attention, we experiment with concatenating the attention information to the neural sentence representations. The final hidden layer then has this form:

$$h = \tanh(W_1 a + W_2 r + b)$$

with r being either the CNN pooling result or the last hidden state of the RNN (see Figure 3).

4 Experimental Setup and Results

4.1 Task and Setup

We evaluate on the two corpora of the CoNLL2010 hedge cue detection task (Farkas

	Model	wiki	bio
(1)	Baseline SVM	62.01*	78.64*
(2)	Baseline RNN	59.82*	84.69
(3)	Baseline CNN	64.94	84.23

Table 1: F_1 results for UD. Baseline models without attention. * indicates significantly worse than best model (in bold).⁴

	Model	wiki	bio
(2)	Baseline RNN	59.82*	84.69
(4)	RNN attention-only	62.02*	85.32
(5)	RNN combined	58.96*	84.88
(3)	Baseline CNN	64.94*	84.23
(6)	CNN attention-only	53.44*	82.85
(7)	CNN combined	66.49	84.69

Table 2: F_1 results for UD. Attention-only vs. combined architectures. Sequence-agnostic weighted average for attention. * indicates significantly worse than best model (bold).

et al., 2010): Wikipedia (11,111 sentences in train, 9634 in test) and Biomedical (14,541 train, 5003 test). It is a binary sentence classification task. For each sentence, the model has to decide whether it contains uncertain information.

For hyperparameter tuning, we split the training set into core-train (80%) and dev (20%) sets; see appendix for hyperparameter values. We use 400 dimensional word2vec (Mikolov et al., 2013) embeddings, pretrained on Wikipedia, with a special embedding for unknown words.

For evaluation, we apply the official shared task measure: F_1 of the uncertain class.

4.2 Baselines without Attention

Our baselines are a support vector machine (SVM) and two standard neural networks without attention, an RNN and a CNN. The SVM is a reimplementation of the top ranked system on Wikipedia in the CoNLL-2010 shared task (Georgescul, 2010), with parameters set to Georgescul (2010)’s values; it uses bag-of-word (BOW) vectors that only include hedge cues. Our reimplementation is slightly better than the published result: 62.01 vs. 60.20 on wiki, 78.64 vs. 78.50 on bio.

The results of the baselines are given in Table 1. The CNN (line 3) outperforms the SVM (line 1) on both datasets, presumably because it considers all words in the sentence – instead of only predefined hedge cues – and makes effective use of this additional information. The RNN (line 2) performs better than the SVM and CNN on biomedical data,

⁴randomization test with $p < .05$.

but worse on Wikipedia. In Section 5.2, we investigate possible reasons for that.

4.3 Experiments with Attention Mechanisms

For the first experiments of this subsection, we use the sequence-agnostic weighted average for attention (see Eq. 10), the standard in prior work.

Attention-only vs. Combined Architecture.

For the case of internal attention, we first remove the final pre-output layer of the standard RNN and the standard CNN to evaluate attention-only architectures. This architecture works well for RNNs but not for CNNs. The CNNs achieve better results when the pooling output (unweighted selection) is combined with the attention output (weighted selection). See Table 2 for F_1 scores.

The baseline RNN has the difficult task of remembering the entire sentence over long distances – the attention mechanism makes this task much easier. In contrast, the baseline CNN already has an effective mechanism for focusing on the key parts of the sentence: k-max pooling. Replacing k-max pooling with attention decreases the performance in this setup.

Since our main goal is to explore the benefits of adding attention to existing architectures (as opposed to developing attention-only architectures), we keep the standard pre-output layer of RNNs and CNNs in the remaining experiments and combine it with the attention layer as in Figure 3.

Focus and Source of Attention. We distinguish different focuses and sources of attention. For focus, we investigate two possibilities: the input to the network, i.e., word embeddings (F=W); or the hidden representations of the RNN or CNN (F=H). For source, we compare internal (S=I) and external attention (S=E). This gives rise to four configurations: (i) internal attention with focus on the first layer of the standard RNN/CNN (S=I, F=H), see lines (5) and (7) in Table 2, (ii) internal attention with focus on the input (S=I, F=W), (iii) external attention on the first layer of RNN/CNN (S=E, F=H) and (iv) external attention on the input (S=E, F=W). The results are provided in Table 3.

For both RNN (8) and CNN (13), the best result is obtained by focusing attention directly on the word embeddings.⁵ These results suggest that it is best to optimize the attention mechanism directly on the input, so that information can be extracted

⁵The small difference between the RNN results on bio on lines (5) and (8) is not significant.

	Model	S	F	wiki	bio
(2)	Baseline RNN	-	-	59.82*	84.69
(5)	RNN combined	I	H	58.96*	84.88
(8)	RNN combined	I	W	62.18*	84.81
(9)	RNN combined	E	H	61.19*	84.62
(10)	RNN combined	E	W	61.87*	84.41
(3)	Baseline CNN	-	-	64.94*	84.23*
(7)	CNN combined	I	H	66.49	84.69
(11)	CNN combined	I	W	65.13*	84.99
(12)	CNN combined	E	H	64.14*	84.73
(13)	CNN combined	E	W	67.08	85.57

Table 3: F_1 results for UD. Focus (F) and source (S) of attention: Internal (I) vs external (E) attention; attention on word embeddings (W) vs. on hidden layers (H). Sequence-agnostic weighted average for attention. * indicates significantly worse than best model (bold).

that is complementary to the information extracted by a standard RNN/CNN.

For focus on input (F=W), external attention (13) is significantly better than internal attention (11) for CNNs. Thus, by designing an architectural element – external attention – that makes it easier to identify hedge cue properties of words, the learning problem is apparently made easier.

For the RNN and F=W, external attention (10) is not better than internal attention (8): results are roughly tied for bio and wiki. Perhaps the combination of the external resource and the more indirect representation of the entire sentence produced by the RNN is difficult. In contrast, hedge cue patterns identified by convolutional filters of the CNN can be evaluated well based on external attention; e.g., if there is strong external-attention evidence for uncertainty, then the effect of a hedge cue pattern (hypothesized by a convolutional filter) on the final decision can be boosted.

In summary, the CNN with external attention achieves the best results overall. It is significantly better than the standard CNN that uses only pooling, both on Wikipedia and biomedical texts. This demonstrates that the CNN can make effective use of external information – a lexicon of uncertainty cues in our case.

Sequence-agnostic vs. Sequence-preserving. Commonly used attention mechanisms simply average the vectors in the focus of attention. This means that sequential information is not preserved. We use the term sequence-agnostic for this. In contrast, we propose to investigate sequence-preserving attention as presented in Section 3.3. We expect this to be important for many

	average		k-max sequence	
	all	k-max	per-dim	per-pos
Wiki	67.08	67.52	66.73	66.50
Bio	85.57	84.36	84.05	84.03

Table 4: F_1 results for UD. Model: CNN, S=E, F=W (13). Sequence-agnostic vs. sequence-preserving attention.

NLP tasks. Sequence-preserving attention is similar to k-max pooling which also selects an ordered subset of inputs. While traditional k-max pooling is unweighted, our sequence-preserving ways of attention still make use of the attention weights.

Table 4 compares k-max pooling, attention and two “hybrid” designs, as described in Section 3.3. We run these experiments only on the CNN with external attention focused on word embeddings (Table 3, line 13), the best performing configuration in the previous experiments.

First, we investigate what happens if we “discretize” attention and only consider the values with the top k attention weights. This increases performance on wiki (from 67.08 to 67.52) and decreases it on bio (from 85.57 to 84.36). We would not expect large differences since attention values tend to be peaked, so for common values of k ($k \geq 3$ in most prior work on k-max pooling) we are effectively comparing two similar weighted averages, one in which most summands get a weight of 0 (k-max average) and one in which most summands get weights close to 0 (average over all, i.e., standard attention).

Next, we compare sequence-agnostic with sequence-preserving attention. As described in Section 3.3, two variants are considered. In k-max pooling, we select the k largest weighted values per dimension (per-dim in Table 4). In contrast, k-max sequence (per-pos) selects all values of the k positions with the highest attention weights.

In Table 4, the sequence-preserving architectures are slightly worse than standard attention (i.e., sequence-agnostic averaging), but not significantly: performance is different by about half a point. This shows that k-max sequence and attention can similarly be used to select a subset of the information available, a parallel that has not been highlighted and investigated in detail before.

Although in this case, sequence-agnostic attention is better than sequence-preserving attention, we would not expect this to be true for all tasks. Our motivation for introducing sequence-

Model	wiki	bio
SVM (Georgescul, 2010)	62.01	78.64
HMM (Li et al., 2014)	63.97	80.15
CRF + ling (Tang et al., 2010)	55.05	86.79
Our CNN with external attention	67.52	85.57

Table 5: Comparison of our best model with the state of the art

preserving attention was that the semantic meaning of a sentence can vary depending on where an uncertainty cue occurs. However, the core of uncertainty detection is keyword and keyphrase detection; so, the overall sentence structure might be less important for this task. For tasks with a stronger natural language understanding component, such as summarization or relation extraction, on the other hand, we expect sequences of weighted vectors to outperform averaged vectors. In Section 6, we show that sequence-preserving attention indeed improves results on a sentiment analysis dataset.

4.4 Comparison to State of the Art

Table 5 compares our models with the state of the art on the uncertainty detection benchmark datasets. On Wikipedia, our CNN outperforms the state of the art by more than three points. On bio, the best model uses a large number of manually designed features and an exhaustive corpus preprocessing (Tang et al., 2010). Our models achieve comparable results without preprocessing or feature engineering.

5 Analysis

5.1 Analysis of Attention

In an analysis of examples for which pooling alone (i.e., the standard CNN) fails, but attention correctly detects an uncertainty, two patterns emerge.

In the first pattern, we find that there are many cues that have more words than the filter size (which was 3 in our experiments), e.g., “it is widely expected”, “it has also been suggested”. The convolutional layer of the CNN is not able to detect phrases longer than the filter size while for attention there is no such restriction.

The second pattern consists of cues spread over the whole sentence, e.g., “Observations of the photosphere of 47 Ursae Majoris *suggested* that the periodicity *could not* be explained by stellar activity, making the planet interpretation *more likely*” where we have set the uncertainty cues

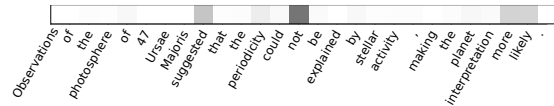


Figure 4: Attention weight heat map

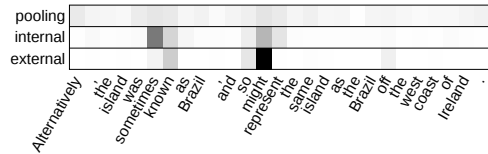


Figure 5: Pooling vs. internal vs. ext. attention

that are distributed throughout the sentence in italics. Figure 4 shows the distribution of external attention weights computed by the CNN for this sentence. The CNN pays the most attention to the three words/phrases “suggested”, “not” and “more likely” that correspond almost perfectly to the true uncertainty cues. K-max pooling of standard CNNs, on the other hand, can only select the k maximum values per dimension, i.e., it can pick at most k uncertainty cues per dimension.

Pooling vs. Internal vs. External Attention.

Finally, we compare the information that pooling, internal and external attention extract. For pooling, we calculate the relative frequency that a value from an n-gram centered around a specific word is picked. For internal and external attention, we directly plot the attention weights α_i . Figure 5 shows the results of the three mechanisms for an exemplary sentence. For a sample of randomly selected sentences, we observed similar patterns: Pooling forwards information from different parts all over the sentence. It has minor peaks at relevant n-grams (e.g. “was sometimes known as” or “so might represent”) but also at non-relevant parts (e.g. “Alternatively” or “the same island”). There is no clear focus on uncertainty cues. Internal attention is more focused on the relevant words. External attention finally has the clearest focus. (See appendix for more examples.)

5.2 Analysis of CNN vs RNN

While the results of the CNN and the RNN are comparable on bio, the CNN clearly outperforms the RNN on wiki. The datasets vary in several aspects, such as average sentence lengths (wiki: 21, bio: 27)⁶, size of vocabularies (wiki: 45.1k,

⁶number of tokens per sentence after tokenization with Stanford tokenizer (Manning et al., 2014).

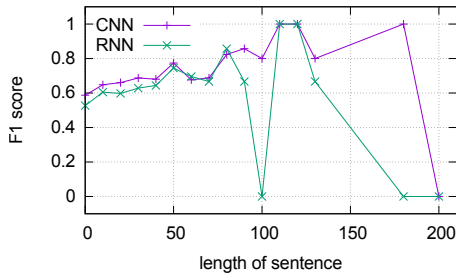


Figure 6: F_1 results for different sentence lengths

bio: 25.3k), average number of out-of-vocabulary (OOV) words per sentence w.r.t. our word embeddings (wiki: 4.5, bio: 6.5), etc. All of those features can influence model performance, especially because of the different way of sentence processing: While the RNN merges all information into a single vector, the CNN extracts the most important phrases and ignores all the rest. In the following, we analyze the behavior of the two models w.r.t. sentence length and number of OOVs.

Figure 6 shows the F_1 scores on Wikipedia of the CNN and the RNN with external attention for different sentence lengths. The lengths have been accumulated, i.e., index 0 on the x-axis includes the scores for all sentences of length $l \in [0, 10)$. Most sentences have lengths $l < 50$. In this range, the CNN performs better than the RNN but the difference is small. For longer sentences, however, the CNN clearly outperforms the RNN. This could be one reason for the better overall performance.

A similar plot for F_1 scores depending on the number of OOVs per sentence does not give additional insights into the model behaviors: The CNN performs better than the RNN independent of the number of OOVs (Figure in appendix).

Another important difference between CNN and RNN is the distribution of precision and recall. While on bio, precision and recall are almost equal for both models, the values vary on wiki:

	P	R
CNN	52.5	85.1
CNN + external attention	58.6	78.3
RNN	75.2	49.6
RNN + external attention	76.3	52.0

Those values suggest that the RNN predicts uncertainty more reluctantly than the CNN.

6 Outlook: Different Task

To investigate whether our attention methods are also applicable to other tasks, we evaluate them

Model	S	F	test set
Baseline CNN	-	-	84.84
CNN attention-only	I	H	83.56
CNN combined	I	H	85.22
CNN combined	I	W	86.11
CNN combined	E	H	86.06
CNN combined	E	W	86.89

Table 6: Accuracy on SST-2, different focus and source of attention.

average		k-max sequence	
all	k-max	per-dim	per-pos
86.89	86.39	87.00	87.22

Table 7: Accuracy on SST-2, sequence-agnostic vs. sequence-preserving attention.

on the 2-class Stanford Sentiment Treebank (SST-2) dataset⁷ (Socher et al., 2013). For a baseline model, we train a CNN similar to our uncertainty CNN but with convolutional filters of different widths, as proposed in (Kim, 2014), and extend it with our attention layer. As cues for external attention, we use the most frequent positive phrases from the train set. Our model is much simpler than the state-of-the-art models for SST-2 but still achieves reasonable results.⁸

The results in Table 6 show the same trends as the CNN results in Table 3, suggesting that our methods are applicable to other tasks as well. Table 7 shows that the benefit of sequence-preserving attention is indeed task dependent. For sentiment analysis on SST-2, sequence-preserving methods outperform the sequence-agnostic ones.

7 Related Work

Uncertainty Detection. Uncertainty has been extensively studied in linguistics and NLP (Kiparsky and Kiparsky, 1970; Karttunen, 1973; Karttunen and Zaenen, 2005), including modality (Saurí and Pustejovsky, 2012; De Marneffe et al., 2012; Szarvas et al., 2012) and negation (Velldal et al., 2012; Baker et al., 2012). Szarvas et al. (2012), Vincze (2014b) and Zhou et al. (2015) conducted cross domain experiments. Domains studied include news (Saurí and Pustejovsky, 2009), biomedicine (Vincze et al., 2008), Wikipedia (Ganter and Strube, 2009) and social media (Wei et al., 2013). Corpora such as FactBank (Saurí and Pustejovsky, 2009) are annotated in detail with respect to perspective, level of factuality and polar-

⁷<http://nlp.stanford.edu/sentiment>

⁸The state-of-the-art accuracy is about 89.5 (Zhou et al., 2016; Yin and Schütze, 2015).

ity. De Marneffe et al. (2012) conducted uncertainty detection experiments on a version of FactBank extended by crowd sourcing. In this work, we use CoNLL 2010 shared task data (Farkas et al., 2010) since CoNLL provides larger train/test sets and the CoNLL annotation consists of only two labels (certain/uncertain) instead of various perspectives and degrees of uncertainty. When using uncertainty detection for information extraction tasks like KB population (Section 1), it is a reasonable first step to consider only two labels.

CNNs. Several studies showed that CNNs can handle diverse sentence classification tasks, including sentiment analysis (Kalchbrenner et al., 2014; Kim, 2014), relation classification (Zeng et al., 2014; dos Santos et al., 2015) and paraphrase detection (Yin et al., 2016). To our knowledge, we are the first to apply them to uncertainty detection.

RNNs. RNNs have mainly been used for sequence labeling or language modeling tasks with one output after each input token (Bengio et al., 2000; Mikolov et al., 2010). Recently, it has been shown that they are also capable of encoding and restoring relevant information from a whole input sequence. This makes them applicable to machine translation (Cho et al., 2014; Bahdanau et al., 2015) and sentence classification tasks (Zhang and Wang, 2015; Hermann et al., 2015; Rocktäschel et al., 2016). In this study, we apply them to UD for the first time and compare their results with CNNs.

Attention has been mainly used for recurrent neural networks (Bahdanau et al., 2015; Rush et al., 2015; Hermann et al., 2015; Rocktäschel et al., 2016; Peng et al., 2015; Yang et al., 2016). We integrate attention into CNNs and show that this is beneficial for uncertainty detection. Few studies in vision integrated attention into CNNs (Stolenga et al., 2014; Xiao et al., 2015; Chen et al., 2015) but this has not been used often in NLP so far. Exceptions are Meng et al. (2015), Wang et al. (2016) and Yin et al. (2016). Meng et al. (2015) used several layers of local and global attention in a complex machine translation model with a large number of parameters. Our reimplementation of their network performed poorly for uncertainty detection (51.51/66.57 on wiki/bio); we suspect that the reason is that Meng et al. (2015)’s training set was an order of magnitude larger than ours. Our approach makes effective use of a much smaller training set. Yin et al. (2016) compared attention based input representations and attention

based pooling. Instead, our goal is to keep the convolutional and pooling layers unchanged and combine their strengths with attention. Allamanis et al. (2016) applied a convolutional layer to compute attention weights. In this work, we concentrate on the commonly used feed forward layers for that. Comparing them to other options, such as convolution, is an interesting direction for future work.

Attention in the literature computes a weighted average with internal attention weights. In contrast, we investigate different strategies to incorporate attention information into a neural network. Also, we propose external attention. The underlying intuition is similar to attention for machine translation, which learns alignments between source and target sentences, or attention in question answering, which computes attention weights based on a question and a fact. However, these sources for attention are still internal information of the network (the input or previous output predictions). Instead, we learn weights based on an external source – a lexicon of cue phrases.

8 Conclusion

In this paper, we presented novel attention architectures for uncertainty detection: external attention and sequence-preserving attention. We conducted an extensive set of experiments with various configurations along different dimensions of attention, including different focuses and sources of attention and sequence-agnostic vs. sequence-preserving attention. For our experiments, we used two benchmark datasets for uncertainty detection and applied recurrent and convolutional neural networks to this task for the first time. Our CNNs with external attention improved state of the art by more than 3.5 F_1 points on a Wikipedia benchmark. Finally, we showed in an outlook that our architectures are applicable to sentiment classification as well. Investigations of other sequence classification tasks are future work. We made our code publicly available for future research (<http://cistern.cis.lmu.de>).

Acknowledgments

Heike Adel is a recipient of the Google European Doctoral Fellowship in Natural Language Processing and this research is supported by this fellowship.

This work was also supported by DFG (SCHU2246/8-2).

References

- Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *Proceedings of The 33rd International Conference on Machine Learning (ICML-16)*, ICML '16, pages 2091–2100, New York City, NY, USA, June. ACM.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR)*, San Diego, California, USA, May.
- Kathryn Baker, Michael Bloodgood, Bonnie J Dorr, Chris Callison-Burch, Nathaniel W Filardo, Christine Piatko, Lori Levin, and Scott Miller. 2012. Use of modality and negation in semantically-informed syntactic MT. *Computational Linguistics*, 38:411–438.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2000. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.
- Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. Abc-cnn: An attention based convolutional neural network for visual question answering. *Technical Report. arXiv:1511.05960*.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- Marie-Catherine De Marneffe, Christopher D Manning, and Christopher Potts. 2012. Did it happen? the pragmatic complexity of veridicality assessment. *Computational linguistics*, 38:335–367.
- Noa P. Cruz Díaz, Maite Taboada, and Ruslan Mitkov. 2016. A machine-learning approach to negation and speculation detection for sentiment analysis. *Journal of the Association for Information Science and Technology (JASIST)*, 67(9):2118–2136.
- Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China, July. Association for Computational Linguistics.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.
- Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The conll-2010 shared task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 1–12, Uppsala, Sweden, July. Association for Computational Linguistics.
- Viola Ganter and Michael Strube. 2009. Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 173–176, Suntec, Singapore, August. Association for Computational Linguistics.
- Maria Georgescu. 2010. A hedgehop over a max-margin framework using hedge cues. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 26–31, Uppsala, Sweden, July. Association for Computational Linguistics.
- Xiaodong He and David Golub. 2016. Character-level question answering with attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1598–1607, Austin, Texas, November. Association for Computational Linguistics.
- Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 1693–1701, Montreal, Canada. MIT Press.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.
- Lauri Karttunen and Annie Zaenen. 2005. Veridicity. In Graham Katz, James Pustejovsky, and Frank Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in

- Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
- Lauri Karttunen. 1973. Presuppositions of compound sentences. *Linguistic Inquiry*, 4:169–196.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.
- Paul Kiparsky and Carol Kiparsky. 1970. Fact. *Progress in Linguistics*, pages 143–173.
- Xiujun Li, Wei Gao, and Jude W Shavlik. 2014. Detecting semantic uncertainty by learning hedge cues in sentences using an hmm. In *SIGIR 2014 Workshop on Semantic Matching in Information Retrieval*, Queensland, Australia, July.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.
- Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding source language with convolutional neural network for machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 20–30, Beijing, China, July. Association for Computational Linguistics.
- Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, Makuhari, Chiba, Japan, September.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at 1st International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA, May.
- Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. 2015. Towards neural network-based reasoning. In *arXiv preprint arXiv:1508.05508*.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May.
- Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September. Association for Computational Linguistics.
- Roser Saurí and James Pustejovsky. 2009. Factbank: A corpus annotated with event factuality. *Language Resources and Evaluation*, 43:227–268.
- Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? Assessing the factuality degree of events in text. *Computational Linguistics*, 38:261–299.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Marijn F Stollenga, Jonathan Masci, Faustino Gomez, and Jürgen Schmidhuber. 2014. Deep networks with internal selective attention through feedback connections. In *NIPS’14 Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3545–3553, Montreal, Canada, December.
- György Szarvas, Veronika Vincze, Richárd Farkas, György Móra, and Iryna Gurevych. 2012. Cross-genre and cross-domain detection of semantic uncertainty. *Computational Linguistics*, 38:335–367.
- Buzhou Tang, Xiaolong Wang, Xuan Wang, Bo Yuan, and Shixi Fan. 2010. A cascade method for detecting hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 13–17, Uppsala, Sweden, July. Association for Computational Linguistics.
- Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational linguistics*, 38:369–410.
- Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC bioinformatics*, 9:1–9.
- Veronika Vincze. 2014a. Uncertainty detection in hungarian texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1844–1853, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

- Veronika Vincze. 2014b. *Uncertainty Detection in Natural Language Texts*. Ph.D. thesis, University of Szeged.
- Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, Berlin, Germany, August. Association for Computational Linguistics.
- Zhongyu Wei, Junwen Chen, Wei Gao, Binyang Li, Lanjun Zhou, Yulan He, and Kam-Fai Wong. 2013. An empirical study on uncertainty identification in social media context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 58–62, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. 2015. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 842–850, Boston, MA, USA, June.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June. Association for Computational Linguistics.
- Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 204–214, Beijing, China, July. Association for Computational Linguistics.
- Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.
- Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *Technical Report. arXiv:1508.01006*.
- Huiwei Zhou, Huan Yang, Long Chen, Zhenwei Liu, Jianjun Ma, and Degen Huang. 2015. Combining feature-based and instance-based transfer learning approaches for cross-domain hedge detection with multiple sources. In *Social Media Processing*, volume 568, pages 225–232. Springer.
- Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495, Osaka, Japan, December. The COLING 2016 Organizing Committee.

A Supplementary Material

A.1 Parameter Tuning

All parameters and learning rate schedule decisions are based on results on the development set (20% of the official training set). After tuning the hyperparameters (see Tables 8 and 9), the networks are re-trained on the whole training set.

We trained the CNNs with stochastic gradient descent and a fixed learning rate of 0.03. For the RNNs, we used Adagrad (Duchi et al., 2011) with an initial learning rate of 0.1. For all models, we used mini-batches of size 10 and applied L2 regularization with a weight of $1e-5$. To determine the number of training epochs, we looked for epochs with peak performances on the development set.

	Model	# conv filters	filter width	# hidden units	# att hidden units
CNN wiki	(3)	200	3	200	-
	(6)	100	3	500	-
	(7)	200	3	200	-
	(11)	200	3	200	-
	(12)	200	3	200	200
	(13)	100	3	200	200
CNN bio	(3)	200	3	500	-
	(6)	100	3	200	-
	(7)	100	3	500	-
	(11)	200	3	200	-
	(12)	200	3	500	100
	(13)	200	3	50	100

Table 8: Result of parameter tuning for CNN (“att hidden units” is the number of units in the hidden layer of the attention component); Model numbers refer to numbers in the main paper

	Model	# rnn hidden units	# hidden units	# att hidden units
RNN wiki	(2)	10	100	-
	(4)	10	100	-
	(5)	10	200	-
	(8)	10	100	-
	(9)	30	200	200
	(10)	10	200	100
RNN bio	(2)	10	500	-
	(4)	10	500	-
	(5)	10	50	-
	(8)	10	50	-
	(9)	30	100	200
	(10)	10	50	200

Table 9: Result of parameter tuning for RNN

A.2 Additional Examples: Attention Weights

Figure 7 and Figure 8 compare pooling, internal attention and external attention for randomly

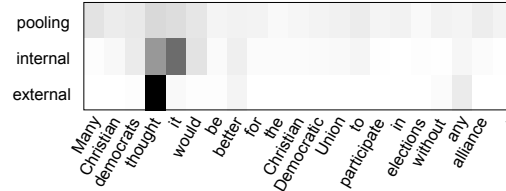


Figure 7: Pooling vs. internal attention vs. external attention

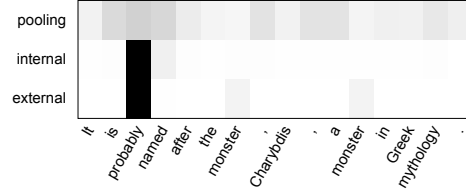


Figure 8: Pooling vs. internal attention vs. external attention

picked examples from the test set. Again, pooling extracts values from all over the sentence while internal and external attention learn to focus on words which can indicate uncertainty (e.g. “thought” or “probably”).

A.3 Additional Figure for Analysis: Results Depending on Number of OOVs

Figure 9 plots the F_1 scores of the CNN and RNN with external attention w.r.t. the number of out-of-vocabulary (OOV) words in the sentences. The number of OOVs have been accumulated, i.e., index 0 on the x-axis includes the score for all sentences with a number of OOVs in $[0,10)$, etc.

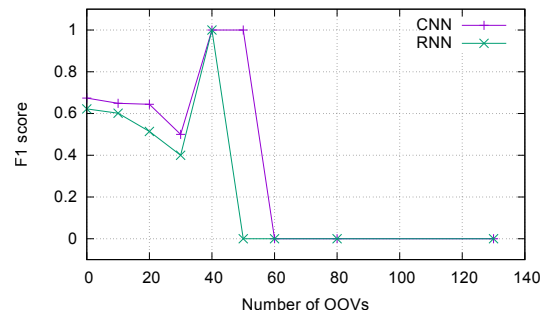


Figure 9: F_1 results for different numbers of OOVs in sentence

Classifying Illegal Activities on Tor Network Based on Web Textual Contents

Mhd Wesam Al Nabki^{1,2}, Eduardo Fidalgo^{1,2}, Enrique Alegre^{1,2}, and Ivan de Paz^{1,2}

¹Department of Electrical, Systems and Automation, University of León, Spain

² Researcher at INCIBE (Spanish National Cybersecurity Institute), León, Spain
{mnab, eduardo.fidalgo, ealeg, ivan.paz.centeno}@unileon.es

Abstract

The freedom of the Deep Web offers a safe place where people can express themselves anonymously but they also can conduct illegal activities. In this paper, we present and make publicly available¹ a new dataset for Darknet active domains, which we call it "Darknet Usage Text Addresses" (DUTA). We built DUTA by sampling the Tor network during two months and manually labeled each address into 26 classes. Using DUTA, we conducted a comparison between two well-known text representation techniques crossed by three different supervised classifiers to categorize the Tor hidden services. We also fixed the pipeline elements and identified the aspects that have a critical influence on the classification results. We found that the combination of TF-IDF words representation with Logistic Regression classifier achieves 96.6% of 10 folds cross-validation accuracy and a macro F1 score of 93.7% when classifying a subset of illegal activities from DUTA. The good performance of the classifier might support potential tools to help the authorities in the detection of these activities.

1 Introduction

If we think about the web as an ocean of data, the Surface Web is no more than the slight waves that float on the top. While in the depth, there is a lot of sunken information that is not reached by the traditional search engines. The web can be divided into Surface Web and Deep Web. The Surface Web is the portion of the web that can be crawled and

¹The dataset is available upon request to the first author (email).

indexed by the standard search engines, such as Google or Bing. However, despite their existence, there is still an enormous part of the web remained without indexing due to its vast size and the lack of hyperlinks, i.e. not referenced by the other web pages. This part, that can not be found using a search engine, is known as Deep Web (Noor et al., 2011; Boswell, 2016). Additionally, the content might be locked and requires human interaction to access e.g. to solve a CAPTCHA or to enter a log-in credential to access. This type of web pages is referred to as "database-driven" websites. Moreover, the traditional search engines do not examine the underneath layers of the web, and consequently, do not reach the Deep Web. The Darknet, which is also known as Dark Web, is a subset of the Deep Web. It is not only not indexed and isolated, but also requires a specific software or a dedicated proxy server to access it. The Darknet works over a virtual sub-network of the World Wide Web (WWW) that provides an additional layer of anonymity for the network users. The most popular ones are "The Onion Router"² also known as Tor network, "Invisible Internet Project" I2P³, and Freenet⁴. The community of Tor refers to Darknet websites as "Hidden Services" (HS) which can be accessed via a special browser called Tor Browser⁵.

A study by Bergman et al. (2001) has stated astonishing statistics about the Deep Web. For example, only on Deep Web there are more than 550 billion individual documents comparing to only 1 billion on Surface Web. Furthermore, in the study of Rudesill et al. (2015) they emphasized on the immensity of the Deep Web which was estimated to be 400 to 500 times wider than the Surface Web.

The concepts of Darknet and Deep Net have ex-

²www.torproject.org

³www.geti2p.net

⁴www.freenetproject.org

⁵www.torproject.org/projects/torbrowser.html.en

isted since the establishment of World Wide Web (WWW), but what make it very popular in the recent years is when the FBI had arrested Dread Pirate Roberts, the owner of Silk Road black market, in October 2013. The FBI has estimated the sales on Silk Road to be 1.2 Billion dollars by July 2013. The trading network covered among 150,000 anonymous customers and approximately 4,000 vendors (Rudesill et al., 2015). The cryptocurrency (Nakamoto, 2008) is a hot topic in the field of Darknet since it anonymizes the financial transactions and hides the trading parties identities (Ron and Shamir, 2014).

The Darknet is often associated with illegal activities. In a study carried out by Intelliagg group (2015) over 1K samples of hidden services, they claimed that 68% of Darknet contents would be illegal. Moore et al. (2016) showed, after analyzing 5K onion domains, that the most common usages for Tor HS are criminal and illegal activities, such as drugs, weapons and all kind of pornography.

It is worth to mention about dramatic increase in the proliferation of Darknet domains which doubled their size from 30K to 60K between August 2015 and 2016 (Figure 1). However, the publicly reachable domains are no more than 6K to 7K due to the ambiguity nature of the Darknet (Ciancaglioni et al., 2016).

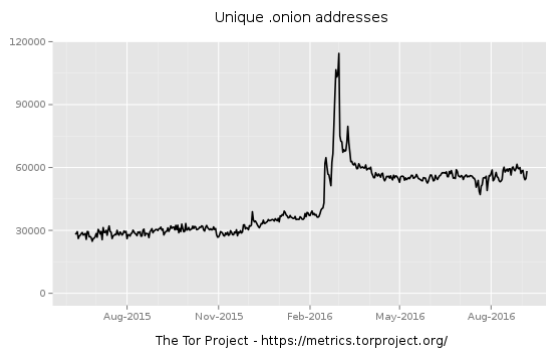


Figure 1: The number of unique *.onion addresses in Tor network between August 2015 to August 2016

Motivated by the critical buried contents on the Darknet and its high abuse, we focused our research in designing and building a system that classifies the illegitimate practices on Darknet. In this paper, we present the first publicly available dataset called "Darknet Usage Text Addresses" (DUTA) that is extracted from the Tor HS Darknet.

DUTA contains 26 categories that cover all the legal and the illegal activities monitored on Darknet during our sampling period. Our objective is to create a precise categorization of the Darknet via classifying the textual content of the HS. In order to achieve our target, we designed and compared different combinations of some of the most well-known text classification techniques by identifying the key stages that have a high influence on the method performance. We set a baseline methodology by fixing the elements of text classification pipeline which allows the scientific community to compare their future research with this baseline under the defined pipeline. The fixed methodology we propose might represent a significant contribution into a tool for the authorities who monitor the Darknet abuse.

The rest of the paper is organized as follows: Section 2 presents the related work. Next, Section 3 explains the proposed dataset DUTA and its characteristics. After that, Section 4 describes the set of the designed classification pipelines. Then, in Section 5 we discuss the experiments performed and the results. In Section 6 we describe the technical implementation details and how we employed the successful classifier in an application. Finally, in Section 7 we present our conclusions with a pointing to our future work.

2 Related Work

In the recent years, many researchers have investigated the classification of the Surface Web (Dumas and Chen, 2000; Sun et al., 2002; Kan, 2004; Kan and Thi, 2005; Kaur, 2014), and the Deep Web (Su et al., 2006; Xu et al., 2007; Barbosa et al., 2007; Lin et al., 2008; Zhao et al., 2008; Xian et al., 2009; Khelghati, 2016). However, the Darknet classification literature is still in its early stages and specifically the classification of the illegal activities (Graczyk and Kinningham, 2015; Moore and Rid, 2016).

Kaur (2014) introduced an interesting survey covering several algorithms to classify web content, paying attention to its importance in the field of data mining. Furthermore, the survey included the pre-processing techniques that might help in features selection, like eliminating the HTML tags, punctuation marks and stemming. Kan et al. explored the use of Uniform Resource Locators (URL) in web classification by extracting the features through parsing and segmenting it (Kan,

2004; Kan and Thi, 2005). These techniques can not be applied to Tor HS since the onion addresses are constructed with 16 random characters. However, tools like Scallion⁶ and Shallot⁷ allow Tor users to create customized .onion addresses based on the brute-force technique e.g. Shallot needs 2.5 years to build only 9 customized characters out of 16. Sun et al. (2002) employed Support Vector Machine (SVM) to classify the web content by taking the advantage of the context features e.g. HTML tags and hyperlinks in addition to the textual features to build the feature set.

Regarding the Deep Web classification, Noor et al. (2011) discussed the common techniques that are used for the content extraction from the Deep Web data sources called "Query Probing", which is commonly used for supervised learning algorithms, and "Visible Form Features" (Xian et al., 2009). Su et al. (2006) have proposed a combination between SVM with query probing to classify the structured Deep Web hierarchically. Barbosa et al. (2007) proposed an unsupervised machine learning clustering pipeline, in which Term Frequency Inverse Document Frequency (TF-IDF) was used for the text representation, and the cosine similarity for distance measurement for the k-means.

With respect to the Darknet, Moore et al. in (2016) have presented a new study based on Tor hidden services to analyze and classify the Darknet. Initially, they collected 5K samples of Tor onion pages and classified them into 12 classes using SVM classifier. Graczyk et al. (2015) proposed a pipeline to classify the products of a famous black market on Darknet, called Agora, into 12 classes with 79% of accuracy. Their pipeline architecture uses the TF-IDF for text features extraction, the PCA for features selection, and SVM for features classification.

Several attempts in literature have been proposed to detect illegal activities whether on the World Wide Web (WWW) network (Biryukov et al., 2014; Graczyk and Kinningham, 2015; Moore and Rid, 2016), peer-to-peer networks (P2P) (Latapy et al., 2013; Peersman et al., 2014) and in chatting messaging systems (Morris and Hirst, 2012). Latapy et al. (2013) investigated P2P systems, e.g. eDonkey, to quantify the paedophile activity by building a tool to detect child-pornography queries

by performing a series of lexical text processing. They found that 0.25% of entered queries are related to pedophilia context, which means that 0.2% of eDonkey network users are entering such queries. However, this method is based on a predefined list of keywords which can not detect new or previously unknown words.

3 The Dataset

3.1 Dataset Building Procedure

To best of our knowledge, there is no labeled dataset that encompasses the activities on the Darknet web pages. Therefore, we have created the first publicly available Darknet dataset and we called it *Darknet Usage Text Addresses (DUTA)* dataset. Currently, DUTA contains only Tor hidden services (HS). We built a customized crawler that utilizes Tor socket to fetch onion web pages through port 80 only i.e. the HTTP protocol. The crawler has 70 worker threads in parallel to download the HTML code behind the HS. Each thread dives into the second level in depth for each HS in order to gather as much text as possible rather than just the index page as in others work (Biryukov et al., 2014). It searches for the HS links on several famous Darknet resources like onion.city⁸ and ahmia.fi⁹. We reached more than 250K HS addresses, but only 7K were alive, and the others were down or not responding. After that, we concatenated the HTML pages of every HS into a single HTML file resulting a single HTML file for each single HS domain. We collected 7,931 hidden services by running the crawler for two months between May and July 2016. For the time being, we labeled 6,831 samples.

3.2 Dataset Characteristics

Darknet researchers have analyzed the HS contents and categorized them into a different number of categories. Biryukov et al. (2014) sampled 1,813 HS and detected 18 categories. Intelliagg group in (2015) analyzed 1K HS samples and classified them into 12 categories. Moore et al. (2016) studied 5,615 HS examples and categorized them into 12 classes. Based on our objective to build a multipurpose dataset and for the sake of completeness, we classified DUTA manually into 26 classes. To the best of our knowledge, this classification is the most extent and complete up to

⁶www.github.com/lachesis/scallion

⁷www.github.com/katmagic/Shallot

⁸www.onion.city

⁹www.ahmia.fi

date. The collected samples were divided among the four authors and each one labeled their designated part; if an author hesitated, it was openly discussed with the rest of the authors. Finally, to check the consistency of the manual labeling, the first author reviewed the final labeling by analyzing random samples of the categorization made by the others.

In addition to labeling the main classes, we dived into labeling the sub-classes of the HS. For example, the class *Counterfeit Personal Identification* has three sub-classes: *Identity Card*, *Driving License*, and *Passport*. Table 1 enumerates DUTA classes.

Main Class	Sub-Class	Count	Main Class	Count	
Violence	Hate	4	Art/ Music	8	
	Hitman	11	Casino/ Gambling	26	
	Weapons	47	Services	285	
Counterfeit Personal Identification	Driving-Licence	4	Cryptocurrency	586	
	ID	7	Down	608	
	Passport	37	Empty	1649	
	File-Sharing	111	Forum	104	
Hosting and Software	Folders	63	Hacking	90	
	Search-Engine	38	Wiki	29	
	Server	95	Leaked-Data	12	
	Software	121	Locked	435	
	Directory	142	Personal	405	
	Drugs	Illegal	230	Politics	8
		Legal	9	Religion	6
Marketplace	Black	63	Library/Books	27	
	White	67	Fraud	4	
Pornography	Child-pornography	914 ⁽¹⁰⁾	Counterfeit Money	55	
	General-pornography	83	Counterfeit Credit Cards	240	
Social-Network	Blog	71	Human-Trafficking	2	
	Chat	47			
	Email	56			
	News	32	The total count	6831	

Table 1: DUTA dataset classes

Counterfeit is a wide class so we split it into three main classes 1) *Counterfeit Personal Identification* which is related to government documents forging. 2) *Counterfeit Money* includes currencies forging and 3) *Counterfeit Credit Cards* covers cloning credit cards, hacked PayPal accounts and fake markets cards like Amazon and eBay. The class *Services* contains the legal services that are provided by individuals or organizations. The class *Down* contains the errors that were returned by the down web pages while crawling them e.g. an SQL error in a website database or a javascript error.

We assign class *Empty* to a web page when:

¹⁰This class includes 57 unique sample plus 857 samples that are extracted from a single forum (See Section 3.2)

1) The text is very short i.e. less than 5 words, 2) It has only images with no text, 3) It contains unreadable text like special characters, numbers, or unreadable words, 4) The empty Cryptolockers pages (ransomware) (Ciancaglini et al., 2016). The class *Locked* contains the HS that require solving a CAPTCHA or a log-in credential. We noticed that some people love to present their works, projects, or even their personal information through an HS page so we labeled them into class *Personal*. The pages that fell under more than one category were labeled based on its main content. For example, we assign *Forum* label to the multi-topic forums unless the whole forum is related to a single topic. e.g. a hacking forum was assigned to *Hacking* class instead of *Forum*. The class *Marketplace* was divided into *Black* when it contained a group of illegal services like Drugs, Weapons, and Counterfeit services and *White* when the marketplace offered legal shops like mobile phones or clothes.

As we have labeled DUTA manually, we realized that some forums on HS contain numerous web pages and all of them are related to a single class i.e. we found a forum about child-pornography that has more than 800 pages of textual content, so we split it up into single samples representing one single forum page, and we added them to the dataset.

4 Methodology

Each classification pipeline is comprised of three main stages. First, text pre-processing, then, features extraction, and finally, classification. We used two famous text representation techniques across three different supervised classifiers resulting six different classification pipelines, and we examined every pipeline to figure out the best combination with the best parameters that can achieve high performance.

4.1 Text Pre-processing

Initially, we eliminated all the HTML tags, and when we detected an image tag, we preserved the image name and removed the extension. Furthermore, we filtered the training set for the non-English samples using Langdetect¹¹ python library and stemmed the text using Porter library from NLTK package¹². Additionally, we re-

¹¹<https://pypi.python.org/pypi/langdetect>

¹²<https://tartarus.org/martin/PorterStemmer/>

moved special characters and stop words thanks to SMART stop list¹³ (Salton, 1971). At this stage, we modified the stop words list by adding 100 words more in order to make it compatible with the work domain. Moreover, we mapped all emails, URLs, and currencies into a single common token for each.

4.2 Features Extraction

After pre-processing the text, we used two famous text representation techniques. A) Bag-of-Words (BOW) is a well-known model for text representation that extracts the features from the text corpus by counting the words frequency. Consequently, every document is represented as a sparse feature vector where every feature corresponds to a single word in the training corpus. B) Term Frequency Inverse Document Frequency model (TF-IDF) (Aizawa, 2003) is a statistical model that assign weights for the vocabularies where it emphasizes the words that occur frequently in a given document, while at the same time de-emphasizes words that occur frequently in many documents. However, even though the BOW and TF-IDF do not take into considerations the words order, they are simple, computationally efficient and compatible with medium dataset sizes.

4.3 Classifier Selection

For each features representation method, we examined three different supervised machine learning algorithms which are Support Vector Machine (SVM) (Suykens and Vandewalle, 1999), Logistic Regression (LR) (Hosmer Jr and Lemeshow, 2004), and Naive Bayes (NB) (McCallum et al., 1998).

5 Empirical Evaluation

5.1 Experimental Setting

Due to the purpose of this paper to classify the Darknet illegal activities, we selected a subset of our DUTA dataset by creating eight categories trying to cover the most representative illegal activities on the Darknet. Another condition that we imposed was that each class in the selected subset should be monotopic (i.e. related to a single category) and contain a sufficient amount of samples (i.e. 40 samples minimum). The rest of the classes are assigned to a 9th category which we

¹³<http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/>

called *Others*. Since we are working on classifying the illegal activities, we did not consider the class *Black-Market* in the training set because its contents are related to more than one class at a single time, and we wanted the classifier to learn from pure patterns. Moreover, when a sample contains relevant images but an irrelevant text or without any textual information, we excluded it from the dataset. Therefore, we had 5,635 samples distributed over nine classes i.e. the eight classes plus the *Others* one (Table 2). After the text pre-processing, we got 5,002 sample split it into a training set that contains 3,501 samples and a testing set of 1,501 samples.

Experiment Main Class	Count
Pornography	963
Cryptocurrency	578
Counterfeit Credit Cards	209
Drugs	169
Violence	60
Hacking	57
Counterfeit Money	46
Counterfeit Personal Identification (Driving-License, ID, Passport)	40
Others	3513

Table 2: Illegal activities dataset classes (A portion of DUTA dataset)

The dataset is highly unbalanced since the largest class has 3,513 samples while the smallest one has only 40 samples. We solved the skew in the dataset thanks to the *class-weight* parameter in Scikit-Learn library¹⁴ which assigns a weight for each class proportional to the number of samples it has (Hauck, 2014). In addition to adjusting the weights of classes, we split up forums by the discussion page (See Section 3.2).

For the models tuning, we applied a grid search over different combinations of parameters with a cross-validation of 10 folds. The successful combination, which corresponds to the selected classification pipeline, is the one that can achieve the highest value of an averaged F1 score metric and an accuracy of 10 folds cross-validation.

We used Python3 with Scikit-Learn machine learning library for the pipelines implementation. We modified the parameters that have a critical influence on the performance of the models. For the BOW dictionary, we set it to 30,000 words with a minimum word frequency of 3, and we left the rest of the parameters to default. Regarding the TF-IDF, we set the maximum feature vectors length to

¹⁴<http://scikit-learn.org/>

10,000 and the minimum to 3. With respect to the classifiers parameters, we kept the default setting for the NB. In contrast, for the LR, we modified only the value of the regularization parameter "C" by setting it to 10 with the balanced class-weight flag activated. For the SVM classifier, we set the decision function parameter to one-vs-rest "ovr", kernel to "RBF", "C" parameter to 10e5, balanced classes weights, and the rest were left to default.

5.2 Results and Discussion

Since we are working on an unbalanced multiclass problem, every class has a precision, a recall, and an F1 score. To combine these three values into a single value, we calculated the macro, micro and weighted average for each class as Table 3 shows. We can see that the pipeline of TF-IDF with LR achieves the highest value with a macro F1 score of 93.7% and the highest cross-validation accuracy of 96.6%. The state-of-the-art paper has achieved 94% accuracy on a different dataset that contains 1K samples (Intelliagg, 2015). Additionally, we plot the macro average precision-recall curve for four classifiers (Figure 2). The plot indicates that the pipeline of TF-IDF with LR achieves the highest precision-recall.

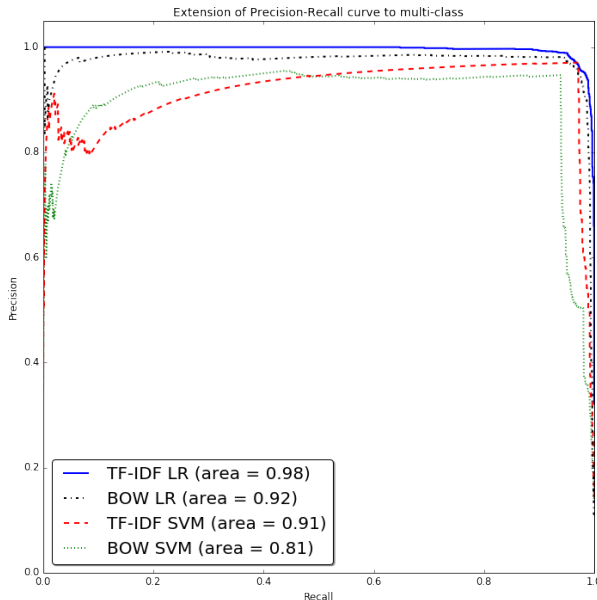


Figure 2: Macro averaging Precision-Recall curve over 4 pipelines, where the area value corresponds to the macro-average Precision-recall curve

Figure 3 shows F1 score comparison between the six classification pipelines over the nine classes. We can see that the classes *Counterfeit*

Metrics/Methods		Average (macro)	Average (micro)	Average (weighted)	CV Accuracy
BOW LR	P	0,952	0,965	0,965	0,958 +/- 0,010
	R	0,889	0,965	0,965	
	F1	0,916	0,965	0,964	
TFIDF LR	P	0,982	0,974	0,975	0,966 +/- 0,010
	R	0,902	0,974	0,974	
	F1	0,937	0,974	0,974	
BOW SVM	P	0,877	0,941	0,942	0,932 +/- 0,013
	R	0,875	0,941	0,941	
	F1	0,874	0,941	0,941	
TFIDF SVM	P	0,983	0,971	0,972	0,960 +/- 0,011
	R	0,882	0,971	0,971	
	F1	0,924	0,971	0,970	
BOW NB	P	0,865	0,941	0,943	0,924 +/- 0,009
	R	0,790	0,941	0,941	
	F1	0,812	0,941	0,940	
TFIDF NB	P	0,530	0,885	0,855	0,863 +/- 0,012
	R	0,425	0,885	0,885	
	F1	0,460	0,885	0,860	

Table 3: A comparison between the classification pipelines with respect to 10 folds cross-validation accuracy (CV), precision (P), recall (R) and F1 score metrics for micro, macro and weighted averaging.

Credit Cards and *Hacking* have a low F1 score over all the pipelines, which is due to several reasons: firstly, the words interference between the classes. For example, the websites which offer counterfeiting credit cards services are most probably "Hack" the credit card system or "Attack" the PayPal accounts, the use sentences like "We hack credit card" or "Hacked Paypal account for sale". Moreover, those classes intersect with *Counterfeit Personal Identification* class due to their similarity from the perspective of forgery. Secondly, the number of samples that were used for training plays an important role during the learning phase, e.g. class *Violence* has 60 samples only.

Nevertheless, the learning curve for the TF-IDF LR pipeline in Figure 4 proves that the algorithm is learning correctly where the validation accuracy curve is raising up and classification accuracy is improving by increasing the number of the samples while the training accuracy curve is starting to decrease slightly. This high accuracy archived will help to build a solid model that will be able to detect illegal activity on Darknet.

6 Application and Implementation

The work presented in the previous sections has been included into an application that can be accessed and tested through a web browser. The implementation of the methods was developed in Python3 using Nltk library to stem the document text, Langdetect library to detect the language of



Figure 3: F1 score comparison for each class for 6 classification pipelines. When a bar is not shown, it means that its value is zero.

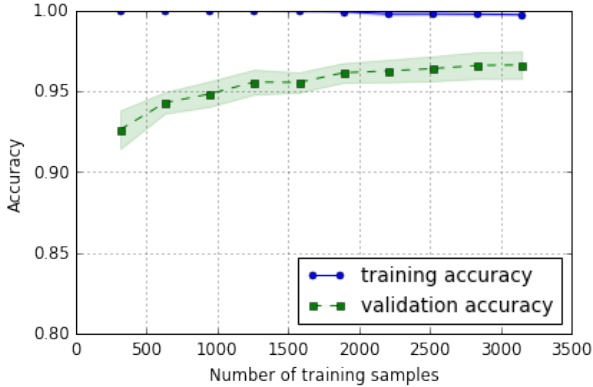


Figure 4: Learning Curve for TF-IDF with LR classifier

the documents and the Scikit-learn library to build the classifiers. The web application is made up of 3 views: one for algorithm selection, the second one for the selection of data to analyze and the third one for showing the results of the analysis (Figure 5).

The Docker image is not publicly available, neither the applications, but under email request, we will grant a temporal access to the web interface.

7 Conclusions and Future Work

In this paper, we have categorized illegal activities of Tor HS by using two text representation methods, TF-IDF and BOW, combined with three classifiers, SVM, LR, and NB. To support the classification pipelines, we built the dataset DUTA,

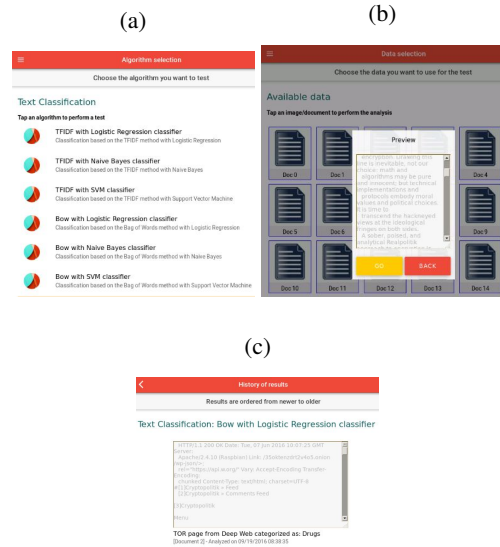


Figure 5: The application has three interfaces. (a) Pipeline selection. (b)The HS content preview. (c) The classification result.

containing 7K samples labeled manually into 26 categories. We picked out nine classes, including the *Others* class, that are related only to illegal activities e.g. drugs trading and child pornography and we used it for training our model. Furthermore, we distinguished the critical aspects that affect the classification pipeline results in term of text representation i.e. the dictionary size and the minimum word frequency influence the text representation techniques performance, and the regularization parameter on the LR and the SVM classifiers. We found that the combination of the TF-IDF text representation with the Logistic Regression classifier can achieve 96.6% accuracy over 10 folds of cross-validation and 93.7% macro F1 score. We noticed that our classifier suffers from overfitting due to the difficulty of reaching more samples of onion hidden services for some classes like counterfeiting personal identification or illegal drugs. However, our results are encouraging, and yet there is still a wide margin for future improvements. We are looking forward to enlarging the dataset by digging deeper into the Darknet by adding more HS sources, even from I2P and Freenet, and exploring ports other than the HTTP port. Moreover, we plan to get the benefit of the HTML tags and the hyperlinks by weighting some tags or parsing the hyperlinks text. Also, during the manual labeling of the dataset, we realized that a wide portion of the hidden services

advertise their illegal products graphically, i.e. the service owner uses the images instead of the text. Therefore, our aim is to build an image classifier to work in parallel with the text classification. The high accuracy we have obtained in this work might represent an opportunity to insert our research into a tool that supports the authorities in monitoring the Darknet.

8 ACKNOWLEDGEMENT

This research was funded by the frame agreement between the University of Len and INCIBE (Spanish National Cybersecurity Institute) under addendum 22. We also want to thanks Francisco J. Rodriguez (INCIBE) for providing us the .onion web pages used to create the dataset.

References

- Akiko Aizawa. 2003. An information-theoretic perspective of tf-idf measures. *Information Processing & Management*, 39(1):45–65.
- Luciano Barbosa, Juliana Freire, and Altigran Silva. 2007. Organizing hidden-web databases by clustering visible web documents. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 326–335. IEEE.
- Michael K. Bergman. 2001. White paper: the deep web: surfacing hidden value. *Journal of electronic publishing*, 7(1).
- Alex Biryukov, Ivan Pustogarov, Fabrice Thill, and Ralf-Philipp Weinmann. 2014. Content and popularity analysis of tor hidden services. In *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 188–193. IEEE.
- Wendy Boswell. 2016. How to mine the invisible web: The ultimate guide.
- V. Ciancaglini, M. Balduzzi, R. McArdle, and M. Rösler. 2016. Below the surface: Exploring the deep web. *Trend Micro Incorporated. As of*, 12.
- Susan Dumais and Hao Chen. 2000. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM.
- Michael Graczyk and Kevin Kinningham. 2015. Automatic product categorization for anonymous marketplaces.
- Trent Hauck. 2014. *scikit-learn Cookbook*. Packt Publishing Ltd.
- David W. Hosmer Jr. and Stanley Lemeshow. 2004. *Applied logistic regression*. John Wiley & Sons.
- Intelliagg. 2015. Deep light shining a light on the dark web. *Magazine*.
- Min-Yen Kan and Hoang Oanh Nguyen Thi. 2005. Fast webpage classification using url features. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 325–326. ACM.
- Min-Yen Kan. 2004. Web page classification without the web page. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 262–263. ACM.
- Prabhjot Kaur. 2014. Web content classification: A survey. *arXiv preprint arXiv:1405.0580*.
- S Mohammadreza Khelghati. 2016. *Deep Web Content Monitoring*. Ph.D. thesis.
- Matthieu Latapy, Clemence Magnien, and Raphael Fournier. 2013. Quantifying paedophile activity in a large p2p system. *Information Processing & Management*, 49(1):248–263.
- Peiguang Lin, Yibing Du, Xiaohua Tan, and Chao Lv. 2008. Research on automatic classification for deep web query interfaces. In *Information Processing (ISIP), 2008 International Symposium on*, pages 313–317. IEEE.
- Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Cite-seer.
- Daniel Moore and Thomas Rid. 2016. Cryptopolitik and the darknet. *Survival*, 58(1):7–38.
- Colin Morris and Graeme Hirst. 2012. Identifying sexual predators by svm classification with lexical and behavioral features. In *CLEF (Online Working Notes/Labs/Workshop)*, volume 12, page 29.
- Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.
- Umara Noor, Zahid Rashid, and Azhar Rauf. 2011. A survey of automatic deep web classification techniques. *International Journal of Computer Applications*, 19(6):43–50.
- Claudia Peersman, Christian Schulze, Awais Rashid, Margaret Brennan, and Carl Fischer. 2014. icop: Automatically identifying new child abuse media in p2p networks. In *Security and Privacy Workshops (SPW), 2014 IEEE*, pages 124–131. IEEE.
- Dorit Ron and Adi Shamir. 2014. How did dread pirate roberts acquire and protect his bitcoin wealth? In *International Conference on Financial Cryptography and Data Security*, pages 3–15. Springer.

- Dakota S Rudesill, James Caverlee, and Daniel Sui. 2015. The deep web and the darknet: A look inside the internet's massive black box. *Woodrow Wilson International Center for Scholars, STIP*, 3.
- Gerard Salton. 1971. The smart retrieval system experiments in automatic document processing.
- Weifeng Su, Jiyang Wang, and Frederick Lochovsky. 2006. Automatic hierarchical classification of structured deep web databases. In *International Conference on Web Information Systems Engineering*, pages 210–221. Springer.
- Aixin Sun, Ee-Peng Lim, and Wee-Keong Ng. 2002. Web classification using support vector machine. In *Proceedings of the 4th international workshop on Web information and data management*, pages 96–99. ACM.
- Johan A.K. Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.
- Xuefeng Xian, Pengpeng Zhao, Wei Fang, Jie Xin, and Zhiming Cui. 2009. Automatic classification of deep web databases with simple query interface. In *Industrial Mechatronics and Automation, 2009. ICIMA 2009. International Conference on*, pages 85–88. IEEE.
- He-Xiang Xu, Xiu-Lan Hao, Shu-Yun Wang, and Yun-Fa Hu. 2007. A method of deep web classification. In *2007 International Conference on Machine Learning and Cybernetics*, volume 7, pages 4009–4014. IEEE.
- Pengpeng Zhao, Li Huang, Wei Fang, and Zhiming Cui. 2008. Organizing structured deep web by clustering query interfaces link graph. In *International Conference on Advanced Data Mining and Applications*, pages 683–690. Springer.

When is multitask learning effective? Semantic sequence prediction under varying data conditions

Héctor Martínez Alonso[♣] Barbara Plank[♡]

[♡] Center for Language and Cognition, University of Groningen, The Netherlands

[♣] Univ. Paris Diderot, Sorbonne Paris Cité – Alpage, INRIA, France

hector.martinez-alonso@inria.fr, b.plank@rug.nl

Abstract

Multitask learning has been applied successfully to a range of tasks, mostly morphosyntactic. However, little is known on *when* MTL works and whether there are data characteristics that help to determine its success. In this paper we evaluate a range of semantic sequence labeling tasks in a MTL setup. We examine different auxiliary tasks, amongst which a novel setup, and correlate their impact to data-dependent conditions. Our results show that MTL is not always effective, significant improvements are obtained only for 1 out of 5 tasks. When successful, auxiliary tasks with compact and more uniform label distributions are preferable.

1 Introduction

The recent success of recurrent neural networks (RNNs) for sequence prediction has raised a great deal of interest, which has lead researchers to propose competing architectures for several language-processing tasks. These architectures often rely on multitask learning (Caruana, 1997).

Multitask learning (MTL) has been applied with success to a variety of sequence-prediction tasks including chunking and tagging (Collobert et al., 2011; Søgaard and Goldberg, 2016; Bjerva et al., 2016; Plank, 2016), name error detection (Cheng et al., 2015) and machine translation (Luong et al., 2016). However, little is known about MTL for tasks which are more *semantic* in nature, i.e., tasks that aim at labeling some aspect of the meaning of words (Cruse, 1986), instead their morphosyntactic behavior. In fact, results on semantic tasks are either mixed (Collobert et al., 2011) or, due to the file drawer bias (Rosenthal, 1979), simply not reported. There is no prior study—to

the best of our knowledge—that compares data-dependent conditions with performance measures to shed some light on when MTL works for semantic sequence prediction. Besides any variation in annotation and conceptualization, the label distributions of such semantic tasks tends to be very different to the characteristic distributions expected in more frequently studied morphosyntactic tasks such as POS-tagging.

The main contribution of this work is an evaluation of MTL on semantic sequence prediction on data-dependent conditions. We derive characteristics of datasets that make them favorable for MTL, by comparing performance with information-theoretical metrics of the label frequency distribution.

We use an off-the-shelf state-of-the-art architecture based on bidirectional Long-Short Term Memory (LSTM) models (Section 3) and evaluate its behavior on a motivated set of main and auxiliary tasks. We gauge the performance of the MTL setup (Section 4) in the following ways: i) we experiment with different combinations of main and auxiliary tasks, using semantic tasks as main task and morphosyntactic tasks as auxiliary tasks; ii) we apply FREQBIN, a frequency-based auxiliary task (see Section 2.5) to a series of language-processing tasks and evaluate its contribution, and iii) for POS we experiment with different data sources to control for label inventory size and corpus source for the auxiliary task.

From our empirical study we observe the MTL architecture’s sensitivity to label distribution properties, and its preference for compact, mid-entropy distributions. Additionally, we provide a novel parametric refinement of the FREQBIN auxiliary task that is more robust. In broader terms, we expect to motivate more thorough analysis of the performance of neural networks in MTL setups.

2 Analyzing multi-task learning

Multitask learning systems are often designed with the intention of improving a *main* task by incorporating joint learning of one or more related *auxiliary* tasks. For example, training a MTL model for the main task of chunking and treating part-of-speech tagging (POS) as auxiliary task.

The working principle of multitask learning is to improve generalization performance by leveraging training signal contained in related tasks (Caruana, 1997). This is typically done by training a single neural network for multiple tasks jointly, using a representation that is shared across tasks. The most common form of MTL is the inclusion of one output layer per additional task, keeping all hidden layers common to all tasks. Task-specific output layers are customarily placed at the outermost layer level of the network.

In the next section, we depict all main and auxiliary tasks considered in this paper.

2.1 Main tasks

We use the following main tasks, aimed to represent a variety of semantic sequence labeling tasks.

FRAMES: We use the FrameNet 1.5 (Baker et al., 1998) annotated corpus for a joint frame detection and frame identification tasks where a word can receive a predicate label like *Arson* or *Personal success*. We use the data splits from (Das et al., 2014; Hermann et al., 2014). While frame identification is normally treated as single classification, we keep the sequence-prediction paradigm so all main tasks rely on the same architecture.

SUPERSENSES: We use the supersense version of SemCor (Miller et al., 1993) from (Ciaramita and Altun, 2006), with coarse-grained semantic labels like *noun.person* or *verb.change*.

NER: The CONLL2003 shared-task data for named entity recognition for labels *Person*, *Loc*, etc. (Tjong Kim Sang and De Meulder, 2003).

SEMTRAITS: We have used the EurWordNet list of ontological types for senses (Vossen et al., 1998) to convert the SUPERSENSES into coarser semantic traits like *Animate* or *UnboundedEvent*.¹

MPQA: The Multi-Perspective Question Answering (MPQA) corpus (Deng and Wiebe, 2015), which contains sentiment information among others. We use the annotation corresponding to the

¹Available at: <https://github.com/bplank/multitasksemantics>

coarse level of annotation, with labels like *attitude* and *direct-speech-event*.

2.2 Auxiliary tasks

We have chosen auxiliary tasks that represent the usual features based on frequency and morphosyntax used for prediction of semantic labels. We collectively refer to them as *lower-level tasks*.

CHUNK: The CONLL2003 shared-task data for noun- and verb-phrase chunking (Tjong Kim Sang and De Meulder, 2003).

DEPREL: The dependency labels for the English Universal Dependencies v1.3 (Nivre et al., 2016).

FREQBIN: The log frequency of each word, treated as a discrete label, cf. Section 2.5.

POS: The part-of-speech tags for the Universal Dependencies v1.3 English treebank.

2.3 Data properties

Table 1 lists the datasets used in this paper, both to train main tasks and auxiliary tasks. For each dataset we list the following metrics: number of sentences, number of tokens, token-type ratio (TTR), the size of the label inventory counting B-labels and I-labels as different ($|Y|$), and the proportion of out-of-span labels, which we refer to as O labels.

The table also provides some of the information-theoretical measures we describe in Section 2.4. Note that DEPRELS and POS are the only datasets without any O labels, while FRAMES and SEMTRAITS are the two tasks with O labels but no B/I-span notation, as tokens are annotated individually.

2.4 Information-theoretic measures

In order to quantify the properties of the different label distributions, we calculate three information-theoretical quantities based on two metrics, kurtosis and entropy.

Entropy is the best-known information-theoretical metric. It indicates the amount of uncertainty in a distribution. We calculate two variants of entropy, one taking all labels in consideration $H(Y_{full})$, and another one $H(Y_{-O})$ where we discard the O label and only measure the entropy for the named labels, such as frame names in FRAMES. The entropy of the label distribution $H(Y_{full})$ is always lower than the entropy for the distribution disregarding the O label $H(Y_{-O})$. This difference is a consequence

	sentences	tokens	TTR	$ Y $	prop of O	k(Y)	$H(Y_{full})$	$H(Y_{-o})$
FRAMES	5.9k	119k	.12	707	.80	701.41	1.60	5.51
MPQA	1.7k	44k	.15	9	.65	2.79	1.12	1.33
NER	22.1k	303k	.10	9	.83	4.10	0.77	1.93
SEMTRAITS	20k	435k	.07	11	.66	5.68	1.29	1.89
SUPERSENSES	20k	435k	.07	83	.66	76.73	1.84	3.53
CHUNK	22.1k	303k	.10	22	.14	3.68	1.73	1.54
DEPRELS	16.6k	255k	.09	47	-	1.80	3.11	3.11
FREQBIN	<i>Same as respective main task</i>			4-7	-	<i>Depends on variant</i>		
POS	16.6k	255k	.09	17	-	-0.20	2.49	2.49

Table 1: Datasets for main tasks (above) and auxiliary tasks (below) with their number of sentences, tokens, type-token ratio, size of label inventory, proportion of O labels, kurtosis of the label distribution, entropy of the label distribution, and entropy of the label distribution without the O label.

of the O-label being often the majority class in span-annotated datasets. The only exception is CHUNK, where O-tokens make up 14% of the total, and the full-distribution entropy is higher.

Kurtosis indicates the skewness of a distribution and provides a complementary perspective to the one given by entropy. The kurtosis of the label distribution describes its tailedness, or lack thereof. The kurtosis for a normal distribution is 3, and higher kurtosis values indicate very tailed distributions, while lower kurtosis values indicate distributions with fewer outliers.

For instance, we can see that larger inventory sizes yield more heavy-tailed distributions, e.g. FRAMES presents a lot of outliers and has the highest kurtosis. The very low value for POS indicates a distribution that, although Zipfian, has very few outliers as a result of the small label set. In contrast, DEPRELS, coming from the same corpus, has about three times as many labels, yielding a distribution that has fewer mid-values while still being less than 3. Nevertheless, the entropy values of POS and DEPRELS are similar, so kurtosis provides a complementary perspective on the data.

2.5 FREQBIN variants

Recently, a simple auxiliary task has been proposed with success for POS tagging: predicting the log frequency of a token (Plank et al., 2016). The intuition behind this model is that the auxiliary loss, predicting word frequency, helps differentiate rare and common words, thus providing better predictions for frequency-sensitive labels. They refer to this auxiliary task as FREQBIN, however, focus on POS only. Plank et al. (2016) used the discretized log frequency of the current word to build the FREQBIN auxiliary task to aid POS

tagging, with good results. This auxiliary task aids the prediction of the main task (POS) in about half the languages, and improves the prediction of out of vocabulary words. Therefore, it is compelling to assess the possible contribution of FREQBIN for other tasks, as it can be easily calculated from the same training data as the main task, and requires no external resources or annotation.

We experiment with three different variants of FREQBIN, namely:

1. SKEWED₁₀: The original formulation of $a = \text{int}(\log_{10}(\text{freqtrain}(w)))$, where a is the frequency label of the word w . Words not in the training data are treated as hapaxes.
2. SKEWED₅: A variant using 5 as logarithm base, namely $a = \text{int}(\log_5(\text{freqtrain}(w)))$, aimed at providing more label resolution, e.g. for the NER data, SKEWED₁₀ yields 4 different labels, and SKEWED₅ yields 6.
3. UNIFORM: Instead of binning log frequencies, we take the index of the k -quantized cumulative frequency for a word w . We use this parametric version of FREQBIN with the median number of labels produced by the previous variants to examine the importance of the label distribution being skewed. For $k=5$, this variant maximizes the entropy of a FREQBIN five-label distribution. Note that this method still places all hapaxes and out-of-vocabulary words of the test data in the same frequency bin.

Even though we could have used a reference corpus to have the same FREQBIN for all the data, we prefer to use the main-task corpus for FREQBIN. Using an external corpus would otherwise lead to a semisupervised learning scenario which is out of the scope of our work. Moreover, in us-

ing only the input corpus to calculate frequency we replicate the setup of Plank et al. (2016) more closely.

3 Model

Recurrent neural networks (RNNs) (Elman, 1990; Graves and Schmidhuber, 2005) allow the computation of fixed-size vector representations for word sequences of arbitrary length. An RNN is a function that reads in n vectors x_1, \dots, x_n and produces a vector h_n , that depends on the entire sequence x_1, \dots, x_n . The vector h_n is then fed as an input to some classifier, or higher-level RNNs in stacked/hierarchical models. The entire network is trained jointly such that the hidden representation captures the important information from the sequence for the prediction task.

A bi-directional recurrent neural network (Graves and Schmidhuber, 2005) is an extension of an RNN that reads the input sequence twice, from left to right and right to left, and the encodings are concatenated. An LSTM (Long Short-Term Memory) is an extension of an RNN with more stable gradients (Hochreiter and Schmidhuber, 1997). Bi-LSTM have recently successfully been used for a variety of tasks (Collobert et al., 2011; Huang et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015; Kiperwasser and Goldberg, 2016; Liu et al., 2015; Plank et al., 2016). For further details, cf. Goldberg (2015) and Cho (2015).

We use an off-the-shelf bidirectional LSTM model (Plank et al., 2016).² The model is illustrated in Figure 1. It is a context bi-LSTM taking as input word embeddings \vec{w} . Character embeddings \vec{c} are incorporated via a hierarchical bi-LSTM using a sequence bi-LSTM at the lower level (Ballesteros et al., 2015; Plank et al., 2016). The character representation is concatenated with the (learned) word embeddings \vec{w} to form the input to the context bi-LSTM at the upper layers. For hyperparameter settings, see Section 3.1.

The stacked bi-LSTMs represent the shared layers between tasks. We here use three stacked ($h=3$) bi-LSTMs for the upper layer, and a single layer bi-LSTM at the lower level for the character representations. Following Collobert et al. (2011), at the outermost ($h = 3$) layer separate output layers for the single tasks are added using a

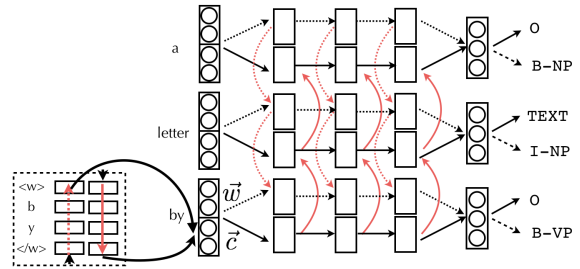


Figure 1: Multi-task bi-LSTM. The input to the model are word \vec{w} and character embeddings \vec{c} (from the lower bi-LSTM). The model is a stacked 3-layer bi-LSTM with separate output layers for the main task (solid line) and auxiliary tasks (dashed line; only one auxiliary task shown in the illustration).

softmax. We additionally experiment with predicting lower-level tasks at inner layers, i.e., predicting POS at $h = 1$, while the main task at $h = 3$, the outermost layer, following Søgaard and Goldberg (2016). During training, we randomly sample a task and instance, and backpropagate the loss of the current instance through the shared deep network. In this way, we learn a joint model for main and auxiliary task(s).

3.1 Hyperparameters

All the experiments in this article use the same bi-LSTM architecture described in Section 3. We train the bi-LSTM model with default parameters, i.e., SGD with cross-entropy loss, no mini-batches, 30 epochs, default learning rate (0.1), 64 dimensions for word embeddings, 100 for character embeddings, 100 hidden states, random initialization for the embeddings, Gaussian noise with $\sigma=0.2$. We use a fixed random seed set upfront to facilitate replicability. The only hyperparameter we further examine is the number of epochs, which is set to 30 unless otherwise specified.

We follow the approach of Collobert et al. (2011) in that we do *not* use any task-specific features beyond word and character information, nor do we use pre-trained word embeddings for initialisation or more advanced optimization techniques.³ While any of these changes would likely improve the performance of the systems, the goal of our experiments is to delimit the behavior of the bi-LSTM architecture and the interaction between main and auxiliary task(s).

²Available at: <https://github.com/bplank/bilstm-aux>

³For example, AdamTrainer or MomentumSGDTrainer in py rnn.

3.2 Experimental Overview

A system in our experiments is defined by a main task and up to two auxiliary tasks, plus a choice of output layers (at which layer to predict the auxiliary task, i.e., $h \in \{1,2,3\}$). For each main task, we ran the following systems:

1. Baseline, without any auxiliary task.
2. One additional system for each auxiliary task, say DEP REL.
3. A combination of each of the three versions of FREQB IN, namely SKEWED₅, SKEWED₁₀ and UNIFORM, and each of the other auxiliary tasks, such as DEP REL+UNIFORM.

The total combination of systems for all five main tasks is 1440.

4 Results

This section describes the results of both experimental scenarios, namely the benchmarking of FREQB IN as an auxiliary task, and the combinations of semantic main task with low-level auxiliary tasks, including an analysis of the data properties. The different tasks in our experiments typically use different evaluation metrics, however we evaluate all tasks on micro-averaged F1 without the O class, which we consider the most informative overall. We do not use the O-label’s F1 score because it takes recall into consideration, and it is deceptively high for the majority class. We test for significance with a 10K-iteration bootstrap sample test, and $p < .05$.

4.1 Main semantic tasks

This section presents the results for the prediction of the main semantic tasks described in Section 2. Given the size of the space of possible task combinations for MTL, we only report the baseline and the results of the best system. Table 2 presents the results for all main semantic tasks, comparing the results of the best system with the baseline. The last column indicates the amount of systems that beat the baseline for a given certain main task. Having fixed the variant of FREQB IN to UNIFORM (see Section 4.2), and the number of epochs to 30 (see below) on development data, the total amount of systems for any main task is 22.

Out of the two main tasks over the baseline only SEM TRAIT S is significantly better over BL. SEM TRAIT S has a small label set, so the system is able to learn shared parameters for the label combinations of main and aux without suffering from too

	BL	Δ Best	Description	aux layer	# over
FRAMES	38.93	-8.13	+FREQB IN	outer	0
MPQA	28.26	0.96	+POS+FREQB IN	inner	2
NER	90.60	-0.58	+FREQB IN	inner	0
SEM TRAIT S	70.42	<u>1.24</u>	+FREQB IN	outer	13
SUPERSENSES	62.36	-0.13	+POS+FREQB IN	inner	0

Table 2: Baseline (BL) and best system performance difference (Δ) for all main tasks—improvements in bold, significant improvements underlined—plus number of systems over baseline for each main task.

much sparsity. Compare with the dramatic loss of the already low-performing FRAMES, which has the highest kurtosis caused by the very long tail of low-frequency labels.

We have expected CHUN K to aid SUPERSENSES, but in spite of our expectations, other low-level tasks do not aid in general the prediction of high-level task. What is otherwise an informative feature for a semantic task in single-task learning does not necessarily lend itself as an equally useful auxiliary task for MTL.

For a complementary evaluation, we have also measured the precision of the O label. However, precision score is also high, above 90, for all tasks except the apparently very difficult MPQA (70.41 for the baseline). All reported systems degrade around 0.50 points with regards to the baseline, except SUPERSENSES which improves slightly from 96.27 to 96.44. The high precision obtained for the also very difficult FRAMES tasks suggests that this architecture, while not suitable for frame disambiguation, can be used for frame-target identification. Disregarding FREQB IN, the only low-level tasks that seems to aid prediction is POS.

An interesting observation from the BIO task analysis is that while the standard bi-LSTM model used here does not have a Viterbi-style decoding like more complex systems (Ma and Hovy, 2016; Lample et al., 2016), we have found very few invalid BIO sequences. For NER, there are only ten I-labels after an O-label, out of the 27K predicted by the bi-LSTM. For SUPERSENSES there are 59, out of 1,5K predicted I-labels.

The amount of invalid predicted sequences is lower than expected, indicating that an additional decoding layer plays a smaller role in prediction quality than label distribution and corpus size, e.g. NER is a large dataset with few labels, and the system has little difficulty in learning label precedences. For larger label sets or smaller data sizes,

invalid sequence errors are bound to appear because of sparseness.

Effect of output layer choice We observe no systematic tendency for an output layer to be a better choice, and the results of choosing the inner- or outer-layer ($h=1$ vs $h=3$) input differ only minimally. However, both systems that include POS have a preference for the inner layer having higher performance, which is consistent with the results for POS in (Søgaard and Goldberg, 2016).

Effect of the number of training epochs Besides all the data properties, the only hyperparameter that we examine further is the number of network training epochs.⁴ All the results reported in this article have been obtained in a 30-epoch regime. However, we have also compared system performance with different numbers of epochs. Out of the values we have experimented (5,15,30,50) with, we recommend 30 iterations for this architecture. At 5 and 15 epochs, the performance does not reach the levels for 30 and is consistently worse for baselines and auxiliary-task systems. Moreover, the performance for 50 is systematically worse than for 30, which indicates overfitting at this point.

Effect of training data size We have run all systems increasing the size of the main task training data in blocks of 25%, keeping the size of the auxiliary task constant. We do not observe improvements over baseline along the learning curve for any of the main tasks except MPQA and SEMTRAITS. At smaller main task data sizes, the auxiliary task learning swamps the training of the main task. This result is consistent with the findings by Luong et al. (2016). We leave the research on the effects auxiliary data size—and its size ratio with regards to the main task—for further work.

4.2 Auxiliary task contribution

As follows from the results so far, the bi-LSTM will not benefit from auxiliary loss if there are many labels and entropy is too high. Auxiliary task level distribution also plays a role, as we will discuss in Section 4.3, FREQBIN-UNIFORM consistently outperforms the skewed measure with base 5 and 10.

⁴Number of epochs is among the most influential parameters of the system. Adding more layers did not further improve results.

	BL	Δ UD/UPOS	Δ UD/PTB	Δ WSJ/PTB
FRAMES	38.93	-14.64	-16.02	-28.18
NER	90.60	-1.36	-2.05	-2.56
MPQA	28.26	-5.62	-13.53	-14.81
SEMTRAITS	70.42	0.67	-0.3	-0.14
SUPERSENSES	62.36	-2.86	-2.83	-6.32
CHUNK	94.76	0.2	0.18	0.18
DEPRELS	88.70	-0.19	-0.18	-1.06
POS	94.36	-	0.18	-0.53

Table 3: Comparison different POS variants (data source/tag granularity): Baseline (BL) and the difference in performance on the +POS system when using the UD Corpus with UPOS (UD/UPOS) or with PTB tabs (UD/PTB), as well as the Wall Street Journal with PTB tags (WSJ/PTB).

Therefore we have also measured the effect of using different sources of POS auxiliary data to give account for the possible differences in label inventory and corpus for all tasks, high and low-level, cf. Table 3. The English UD treebank is distributed with Universal POS (UPOS), which we use throughout this article, and also with Penn Treebank (PTB) tags (Marcus et al., 1993). We have used the PTB version of the English UD corpus (UD/PTB) as well as the training section of the Wall Street Journal (WSJ) treebank as of POS (WSJ/PTB) auxiliary task. The former offers the opportunity to change the POS inventory to the three times larger PTB inventory while using the same corpus.

However, the characteristics of the UD/UPOS we have used as POS throughout the article makes it a more suitable auxiliary source, in fact it systematically outperforms the other two. We argue that UD/UPOS has enough linguistic signal to be a useful auxiliary task, while still depending on a smaller label inventory. Interestingly, if we use POS for CHUNK (cf. Table 3), note that even though the language in WSJ is closer to the language in the training corpora for CHUNK and NER, it is not the best auxiliary POS source for either task.

We observe an improvement when using UD/PTB for POS, while using WSJ/PTB worsens the results for this task. We argue that this architecture benefits from the scenario where the same corpus is used to train with two different label sets for POS, whereas using a larger label set and a different corpus does not aid prediction.

4.3 Analyzing FREQBIN

In this section we evaluate the interaction between all tasks and the FREQBIN auxiliary task. For this purpose, we treat all tasks (high- or low-level) as main task, and compare the performance of a single-task baseline run, with a task +FREQBIN setup. We have compared the three versions of FREQBIN (Section 2.5) but we only report UNIFORM, which consistently outperforms the other two variants, according to our expectations.

Table 4 lists all datasets with the size of their label inventory for reference ($|Y|$), as well as the absolute difference in performance between the FREQBIN-UNIFORM system and the baseline (Δ). Systems that beat the baseline are marked in bold.

Following Plank et al. (2016), the FREQBIN system beats the baseline for the POS task. Moreover, it also aids the prediction for SEMTRAITS and MPQA. The better performance of these two systems indicates that this architecture is not necessarily only advisable for lower-level tasks, as long as the datasets have the right data properties.

	$ Y $	BL	ΔU	R^2
FRAMES	707	38.93	-8.13	.00
MPQA	9	28.26	0.44	.09
NER	9	90.60	-1.31	.26
SEMTRAITS	11	70.42	<u>1.12</u>	.44
SUPERSENSES	83	62.36	-0.69	.47
CHUNK	22	94.76	-0.14	.49
POS	17	94.35	0.21	.68
DEPRELS	47	88.70	-0.16	.64

Table 4: Label inventory size ($|Y|$), FREQBIN-baseline absolute difference in performance (Δ)—improvements are in bold, significant improvements are underlined—and coefficient of determination for label-to-frequency regression (R^2).

The improvement of low-level classes is clear in the case of POS. We observe an improvement from 75 to 80 for the X label, mostly made up of low-frequency items. The similarly scattered label INTJ goes from 84 to 87. While no POS label drops in performance on +FREQBIN with regards to the baseline, all the other improvements are of 1 point of less.

4.4 Label-frequency co-informativeness

To supplement the benchmarking of FREQBIN, we estimate how much frequency information is contained in all the linguistic sequence annotations

used in this article. We do so by evaluating the coefficient of determination (R^2) of a linear regression model to predict the log frequency of a word given its surrounding label trigram, which we use as a proxy for sequence prediction. For instance, for ‘the *happy* child’, it would attempt to predict the log-frequency of *happy* given the ‘DET ADJ NOUN’ POS trigram. Note that this model is delexicalized, and only uses task labels because its goal is to determine how much word-frequency information is contained in e.g. the POS sequence. A high R^2 indicates there is a high proportion of the variance of log frequency explained by the label trigram. We use linear regression implemented in `sklearn` with L2 regularization and report the average R^2 of 10-fold cross-validation.

POS is the label set with the highest explanatory power over frequency, which is expectable: determiners, punctuations and prepositions are high-frequency word types, whereas hapaxes are more often closed-class words. DEPRELS sequences contain also plenty of frequency information. Three sequence tasks have similar scores under .50, namely CHUNK, SUPERSENSE and SEMTRAITS. They all have in common that their O class is highly indicative of function words, an argument supported by their similar values of full-distribution entropy. The one with the lowest score out of these three, namely SEMTRAITS is the one with the least grammatical information, as it does not contain part of speech-related labels. The (R^2) is very low for the remaining tasks, and indeed, for FRAMENET it is a very small negative number which rounds up to zero.

While the co-informativeness of FREQBIN with regards to its main task is a tempting explanation, it does not fully explain when it works as an auxiliary task. Indeed, the FREQBIN contribution at handling out-of-vocabulary words seems to only affect POS and SEMTRAITS, while it does not improve DEPRELS, which normally depends on syntactic trees for accurate prediction.

5 Net capacity and contribution of character representation

In this section we alter the network to study the effect of network width and character representations. Multitask learning allows easy sharing of parameters for different tasks. Part of the explanation for the success of multitask learning are related to *net capacity* (Caruana, 1997). Enlarg-

ing a network’s hidden layers reduces generalization performance, as the network potentially learns dedicated parts of the hidden layer for different tasks. This means that the desirable trait of parameter sharing of MTL is lost. To test this property, we train a MTL network for all setups where we increase the size of the hidden layer by a factor k , where k is the number of auxiliary tasks.

Our results confirm that increasing the size of the hidden layers reduces generalization performance. This is the case for all setups. None of the results is better than the best systems in Table 2, and the effective number of systems that outperform the baseline are fewer (FRAMES: 0, MPQA: 2, NER: 0, SEMTRAITS: 9, SUPERSENSES: 0).

Throughout the article we used the default network structure which includes a lower-level bi-LSTM at the character level. However, we hypothesize that the character features are not equally important for all tasks. In fact, if we disable the character features, making the system only depend on word information (cf. Table 5), we observe that two of the tasks (albeit the ones with the overall lowest performance) increase their performance in about 2.5 points, namely MPQA and FRAMES. For the other two tasks we observe drops up to a maximum of 8-points for NER. Character embeddings are informative for NER, because they approximate the well-known capitalization features in traditional models. Character features are not informative for tasks that are more dependent on word identity (like FRAMES), but are indeed useful for tasks where parts of the word can be informative, such as POS or NER.

	BL ($w + c$)	Δ only w
FRAMES	38.93	+2.39
NER	90.60	-8.05
MPQA	28.26	+2.91
SEMTRAITS	70.42	-3.62
SUPERSENSES	62.36	-4.44
CHUNK	94.76	-0.96
DEPRELS	88.70	-1.87
POS	94.36	-3.18

Table 5: Comparison default hierarchical systems using a lower-level bi-LSTM for characters (BL $w + c$) versus system using only words (w).

6 Related Work

Multitask learning has been recently explored by a number of studies, including name error recog-

nition (Cheng et al., 2015), tagging and chunking (Collobert et al., 2011; Plank et al., 2016), entity and relation extraction (Gupta et al., 2016), machine translation (Luong et al., 2016) and machine translation quality estimation including modeling annotator bias (Cohn and Specia, 2013; Shah and Specia, 2016). Most earlier work had in common that it assumed jointly labeled data (same corpus annotated with multiple labels). In contrast, in this paper we evaluate multitask training from distinct sources to address data paucity, like done recently (Kshirsagar et al., 2015; Braud et al., 2016; Plank, 2016).

Sutton et al. (2007) demonstrate improvements for POS tagging by training a joint CRF model for both POS tagging and noun-phrase chunking. However, it is not clear under what conditions multi-task learning works. In fact, Collobert et al. (2011) train a joint feedforward neural network for POS, chunks and NER, and observe only improvements in chunking (similar to our findings, cf. Section 4.2), however, did not investigate data properties of these tasks.

To the best of our knowledge, this is the first extensive evaluation of the effect of data properties and main-auxiliary task interplay in MTL for semantic sequence tasks. The most related work is Luong et al. (2016), who focus on the effect of auxiliary data size (constituency parsing) on the main task (machine translation), finding that large amounts of auxiliary data swamp the learning of the main task. Earlier work related to MTL is the study by Ando and Zhang (2005) who learn many auxiliary task from unlabeled data to aid morphosyntactic tasks.

7 Conclusions and Future Work

We have examined the data-conditioned behavior of our MTL setup from three perspectives. First, we have tested three variants of FREQBIN showing that our novel parametric UNIFORM variant outperforms the previously used SKEWED₁₀, which has a number of labels determined by the corpus size. Second, we examined main-auxiliary task combinations for five semantic tasks and up to two lower-level tasks. We observe that the best auxiliary task is either FREQBIN or FREQBIN+POS, which have low kurtosis and fairly high entropy.

We also explored three sources of POS data as auxiliary task, differing in corpus composition or

label inventory. We observe that the UPOS variant is the most effective auxiliary task for the evaluated architecture. Indeed, UPOS has fewer labels, and also a more compact distribution with lower kurtosis than its PTB counterpart.

While we propose a better variant of FREQBIN (UNIFORM) we conclude that it is not a useful auxiliary task in the general case. Rather, it helps predict low-frequency labels in scenarios where the main task is already very co-informative of word frequency. While log frequency lends itself naturally to a continuous representation so that we could use regression to predict it instead of classification, doing so would require a change of the architecture and, most importantly, the joint loss. Moreover, discretized frequency distributions allow us to interpret them in terms of entropy. Thus, we leave it to future work.

When comparing system performance to data properties, we determine the architecture’s preference for compact, mid-entropy distributions what are not very skewed, i.e., have low kurtosis. This preference explains why the system fares consistently well for a lot of POS experiments but falls short when used for task with many labels or with a very large O majority class. Regarding output layer choice, we have not found a systematic preference for inner or outer-layer predictions for an auxiliary task, as the results are often very close.

We argue strongly that the difficulty of semantic sequence predictions can be addressed as a matter of data properties and not as the antagonistic truism that morphosyntax is easy and semantics is hard. The underlying problems of semantic task prediction have often to do with the skewedness of the data, associated often to the preponderance of the O-class, and a possible detachment from mainly lexical prediction, such as the spans of MPQA.

This paper is only one step towards better understanding of MTL. It is necessarily incomplete, we hope to span more work in this direction. For instance, the system evaluated in this study has no Viterbi-style decoding for sequences. We hypothesize that such extension of the model would improve prediction of labels with strong interdependency, such as BIO-span labels, in particular for small datasets or large label inventories, albeit we found the current system predicting fewer invalid sequences than expected. In future, we would like to extend this work in several directions: comparing different MTL architectures, additional tasks,

loss weighting, and comparing the change of performance between a label set used as an auxiliary task or as a—predicted—feature.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback. Barbara Plank thanks the Center for Information Technology of the University of Groningen for the HPC cluster and Nvidia corporation for supporting her research. Héctor Martínez Alonso is funded by the French DGA project VerDi.

References

- Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.
- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *ACL*.
- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *EMNLP*.
- Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *COLING*.
- Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of rst discourse parsers. In *COLING*.
- Rich Caruana. 1997. Multitask learning. In *Learning to learn*, pages 95–133. Springer.
- Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-Domain Name Error Detection using a Multi-Task RNN. In *EMNLP*.
- Kyunghyun Cho. 2015. Natural Language Understanding with Distributed Representation. *ArXiv*, abs/1511.07916.
- Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*.
- Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *ACL*.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

- D. Alan Cruse. 1986. *Lexical semantics*. Cambridge University Press.
- Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.
- Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *NAACL*.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *ACL*.
- Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.
- Yoav Goldberg. 2015. A Primer on Neural Network Models for Natural Language Processing. *ArXiv*, abs/1510.00726.
- Alex Graves and Jürgen Schmidhuber. 2005. Frame-wise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.
- Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction. In *COLING*.
- Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *ACL*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *TACL*.
- Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *ACL-IJCNLP*.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.
- Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*.
- Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.
- George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.
- Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *ACL*.
- Barbara Plank. 2016. Keystroke dynamics as signal for shallow syntactic parsing. In *COLING*.
- Robert Rosenthal. 1979. The file drawer problem and tolerance for null results. *Psychological bulletin*, 86(3):638.
- Kashif Shah and Lucia Specia. 2016. Large-scale multitask learning for machine translation quality estimation. In *NAACL*.
- Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.
- Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*, pages 142–147. Association for Computational Linguistics.
- Piek Vossen, Laura Bloksma, Horacio Rodriguez, Salvador Climent, Nicoletta Calzolari, Adriana Roventini, Francesca Bertagna, Antonietta Alonge, and Wim Peters. 1998. The eurowordnet base concepts and top ontology. *Deliverable D017 D*, 34:D036.

Learning Compositionality Functions on Word Embeddings for Modelling Attribute Meaning in Adjective-Noun Phrases

Matthias Hartung Fabian Kaupmann Soufian Jebbara Philipp Cimiano
Semantic Computing Group
CITEC, Bielefeld University
{mhartung, fkaupmann, sjebbara, cimiano}@techfak.uni-bielefeld.de

Abstract

Word embeddings have been shown to be highly effective in a variety of lexical semantic tasks. They tend to capture meaningful relational similarities between individual words, at the expense of lacking the capability of making the underlying semantic relation explicit. In this paper, we investigate the *attribute* relation that often holds between the constituents of adjective-noun phrases. We use CBOW word embeddings to represent word meaning and learn a compositionality function that combines the individual constituents into a phrase representation, thus capturing the compositional attribute meaning. The resulting embedding model, while being fully interpretable, outperforms count-based distributional vector space models that are tailored to attribute meaning in the two tasks of attribute selection and phrase similarity prediction. Moreover, as the model captures a generalized layer of attribute meaning, it bears the potential to be used for predictions over various attribute inventories without re-training.

1 Introduction

Attributes such as SIZE, WEIGHT or COLOR are part of the building blocks of representing knowledge about real-world entities or events (Barsalou, 1992). In natural language, formal attributes find their counterpart in attribute nouns which can be used in order to generalize over individual properties, e.g., *big* or *small* in case of SIZE, *blue* or *red* in case of COLOR (Hartung, 2015).

In order to ascribe such properties to entities or events, adjective-noun phrases are a very frequent linguistic pattern. In these constructions, attribute

meaning is conveyed only implicitly, i.e., without being overtly realized at the phrasal surface. Hence, *attribute selection* has been defined as the task of predicting the hidden attribute meaning expressed by a property-denoting adjective in composition with a noun (Hartung and Frank, 2011b), as in the following examples:

- (1) a. *hot summer* → TEMPERATURE
- b. *hot debate* → EMOTIONALITY
- c. *hot soup* → TASTE/TEMPERATURE

Previous work on this task has largely been carried out in distributional semantic models (cf. Hartung (2015) for an overview). In the face of the recent rise of distributed neural representations as a means of capturing lexical meaning in NLP tasks (Collobert et al., 2011; Mikolov et al., 2013a; Pennington et al., 2014), our goal in this paper is to model attribute meaning based on word embeddings. In particular, we use CBOW embeddings of adjectives and nouns (Mikolov et al., 2013a) as underlying word representations and train a compositionality function in order to compute a phrase representation that is predictive of the implicitly conveyed attribute meaning.

In fact, word embeddings (also referred to as *predict models*) have been shown to be highly effective in a variety of lexical semantic tasks (Baroni et al., 2014b), compared to “traditional” distributional semantic models (or *count models*) in the tradition of Harris (1954). However, this finding has been refuted to a certain extent by Levy et al. (2015), stating that much of the perceived superiority of word embeddings is due to hyperparameter optimizations rather than principled advantages. Moreover, the authors found that in many cases, tailoring count models to a particular task at hand is both feasible and beneficial in order to outperform the more generic embeddings.

This sheds light on a definitive plus of count models, viz. their transparency and interpretability in the sense that their semantic similarity ratings can (under certain conditions) be traced back to particular semantic relations, whereas word embeddings typically yield rather vague and diversified similarities (Erk, 2016). Due to this lack in interpretability, word embeddings are not easily interoperable with symbolic lexical resources or ontologies. Thus, we argue that modelling attribute meaning poses an interesting challenge to word embeddings for two reasons: First, being rooted in ontological knowledge, attribute meaning clearly draws on interpretability of the underlying model; second, attribute meaning in adjective-noun phrases is conveyed in compositional processes (cf. Ex. (1)) which are under-researched in the context of word embeddings so far (Manning, 2015).

Our main contributions in this paper are: (i) We demonstrate that word embeddings can be successfully harnessed for attribute selection – a task that requires both compositional and interpretable representations of phrase meaning. (ii) This is achieved via a learned compositionality function f on adjective and noun embeddings that carves out attribute meaning in their compositional phrase meaning. (iii) We show that f captures generalized attribute meaning (cf. Bride et al. (2015)) that abstracts from individual attributes. Thus, after fitting the compositionality function, our model bears the potential of being applied to various application scenarios (e.g., aspect-based sentiment analysis) involving diverse attribute inventories. (iv) We show that the same model also scales to the task of predicting semantic similarity of adjective-noun phrases, which indicates both the robustness of the model and the importance of attribute meaning as a major source of phrase similarity.

2 Related Work

Attribute Learning from Adjectives and Nouns.

Adjective-centric approaches to attribute learning from text date back to Almuhareb (2006) and Cimiano (2006). Bakhshandeh and Allen (2015) present a sequence tagging model in order to extract attribute nouns from adjective glosses in WordNet. Most recently, Petersen and Hellwig (2016) use a clustering approach based on adjective-noun co-occurrences in order to induce clusters of German adjectives that constitute the

value space of an attribute. However, their approach falls short of making the respective attribute explicit.

These approaches have in common that they do not consider the compositional semantics of an adjective in its phrasal context with a noun in order to derive attribute meaning. This is in contrast to Hartung and Frank (2010; 2011b) who frame attribute selection in a distributional count model which (i) encodes adjectives and nouns as distributional word vectors over attributes as shared dimensions of meaning and (ii) uses vector mixture operations in order to compose these word vectors into phrase representations that are predictive of compositional attribute meaning.

Tandon et al. (2014) propose a semi-supervised method for populating a knowledge base with triples of nouns, attributes and adjectives that are acquired from adjective-noun phrases. Being based on label propagation over monosemous adjectives as seeds, their approach depends on a lexical resource providing initial mappings between adjectives and attributes.

The present approach and the work by Hartung and Frank may be considered as pairs of opposites in two respects: First, our model is based on pre-trained CBOW word embeddings for representing adjective and noun meaning. Thus, we do not encode any attribute-specific lexical information explicitly at the level of word representation. Second, we apply function learning in order to empirically induce a compositionality function that is trained to promote aspects of attribute meaning in adjective-noun phrase embeddings.

Compositionality. Modelling compositional processes at the intersection of word and phrase meaning in distributional semantic models has attracted considerable attention in the last years (Erk, 2012). Mitchell and Lapata (2010) have promoted a variety of vector mixture models for the task, which have been criticized for their syntactic agnosticism (Baroni and Zamparelli, 2010; Guevara, 2010).

Focussing on adjective-noun compositionality, the latter authors propose instead to model adjective meaning as matrices encoding linear mappings between noun vectors. These attempts to integrate formal semantic principles in the tradition of Frege (1892) into a distributional framework have been generalized to a “program for compositional distributional semantics” (Baroni et al.,

2014a) that is centered around *functional application* as the general process to model compositionality in semantic spaces, thus emphasizing the insight that different linguistic phenomena require to be modeled in corresponding algebraic structures and composition operators matching these structures (cf. Widdows (2008), Grefenstette and Sadrzadeh (2011), Grefenstette et al. (2014)).

Bride et al. (2015) observe that such composition operators, by being trained on empirical corpus data, can either be tailored to specific lexical types (i.e., individual composition functions for each adjective in the corpus), or designed to capture general compositional processes in syntactic configurations (i.e., a single lexical function for all adjective-noun phrases). In line with these authors, we aim at learning a lexical function which captures attribute meaning in the compositional semantics of adjective-noun phrases, while *generalizing* over individual attributes.

Contrary to distributional count models, there is relatively few work on applying word embeddings to linguistic problems or NLP tasks related to compositionality. Notable exceptions are Socher et al. (2013) for sentiment analysis, as well as Salehi et al. (2015) and Cordeiro et al. (2016) who focus on predicting the degree of compositionality in nominal compounds rather than carving out a particular semantic relation that is expressed in their compositional semantics.

3 Learning Attribute Meaning in Word Embeddings

3.1 Attribute Meaning in Natural Language

Natural language refers to ontological attributes in terms of attribute nouns such as *color*, *size* or *shape* (Guarino, 1992; Löbner, 2013). Therefore, despite remaining mostly implicit in adjective-noun phrases (cf. Ex. (1) above), we hypothesize that attribute meaning can be learned from contextual patterns of attribute nouns in natural language text. This leads us to the assumption that *adjectives, nouns and attributes (via attribute nouns) can be embedded in the same semantic space*.

3.2 Compositional Models of Attribute Meaning

In this work, we aim at a compositional approach to attribute meaning in adjective-noun phrases. As a consequence of the above assumption, our model represents adjectives, nouns and attributes as vec-

tors \vec{a} , \vec{n} and \vec{attr} , respectively, in one and the same embedding space $\mathcal{S} \subseteq \mathbb{R}^d$.

By designing a composition function $f(\vec{a}, \vec{n})$ that produces phrase representations $\vec{p} \in \mathcal{S}$, we can use nearest neighbour search in \mathcal{S} in order to predict the attribute \widehat{attr} that is most likely expressed in the compositional semantics of an adjective-noun phrase p :

$$\widehat{attr} := \arg \max_{attr \in A} \cos(\vec{p}, \vec{attr}) \quad (2)$$

where $\vec{p} = f(\vec{a}, \vec{n})$, \cos denotes cosine vector similarity and A the set of all attributes considered. The compositional functions that we use in this work can be divided into baseline models, largely derived from Mitchell and Lapata (2010), and trainable models.

3.2.1 Baseline Models

Adjective or Noun. The simplest model is to skip any composition and just use the representation of the adjective or the noun as a surrogate: $\vec{p} = \vec{a}$ or $\vec{p} = \vec{n}$, respectively.

Pointwise Vector Addition. The first step in the direction of compositionality is pointwise vector addition: $\vec{p} = \vec{a} + \vec{n}$. According to Mitchell and Lapata (2010), the commutativity of addition is a disadvantage because the model ignores word order and thus syntactic information is lost.

Weighted Vector Addition. For the latter reason, Mitchell and Lapata (2010) also propose a weighted variant of pointwise vector addition. In order to account for possibly different contributions of the constituents to phrasal composition, scalar weights α and β are applied to the word vectors before pointwise addition: $\vec{p} = \alpha\vec{a} + \beta\vec{n}$.

Pointwise Vector Multiplication. This composition function multiplies the individual dimensions of the adjective and noun vector: $p_i = a_i \cdot b_i$. Mitchell and Lapata (2010) point out that vector multiplication can be seen as equivalent to logical intersection. In previous work on attribute selection in a count-based distributional framework, the best results were obtained using pointwise multiplication (Hartung, 2015).

Dilation. The dilation model of Mitchell and Lapata (2010) dilates one vector in the direction of the other. This is inspired by the dilation effect of matrix multiplication, but is specifically designed

to be basis-independent:

$$\vec{p} = (\vec{n} \cdot \vec{n})\vec{a} + (\lambda - 1)(\vec{n} \cdot \vec{a})\vec{a} \quad (3)$$

Here, \vec{n} is stretched by a factor λ to emphasize the contribution of \vec{a} . λ is a parameter that has to be chosen manually. Analogously, dilation of the adjective is possible as well.

3.2.2 Trainable Models

In this section, we present a method for supervised training of compositionality functions. We propose additive and multiplicative models that use weighting matrices or tensors to balance the contributions of adjectives and nouns. The composition is trained to specifically capture attribute meaning in the resulting phrase representation. The weights are trained as part of a shallow neural network (see Section 3.2.3).

Full Weighted Additive Model. Following Guevara (2010), the full additive model capitalizes on vector addition with weighting matrices for adjective and noun:

$$\vec{p} = \mathbf{A} \cdot \vec{a} + \mathbf{N} \cdot \vec{n} \quad (4)$$

As initializations of the weighting matrices, we use an identity matrix¹, which is equivalent to non-parametric vector addition. As weighting schemes, we use one of (i) weighting only the adjective or noun, respectively, or (ii) weighting both adjective and noun distinctly.

Note that, in line with Guevara (2010), this model makes use of weight matrices in order to balance the contribution of adjectives and nouns to phrasal attribute meaning, whereas Mitchell and Lapata (2010) use scalar weights in their pointwise additive model (cf. Section 3.2.1). Our intuition is that full additive models should be better suited to model compositional processes that involve interactions between dimensions of meaning.

Trained Tensor Product. As a weighted multiplicative model, we use multiplication of adjective and noun representations with a learned third-order tensor \mathbf{T} , following Bride et al. (2015):

$$\vec{p} = \vec{a}^T \cdot \mathbf{T}^{[1:d]} \cdot \vec{n} \quad (5)$$

with $\vec{a} \in \mathbb{R}^d$, $\vec{n} \in \mathbb{R}^d$, $\mathbf{T}^{[1:d]} \in \mathbb{R}^{d \times d \times d}$

¹We also experimented with different initializations such as random values, all-ones, or an identity matrix with additional small random values on non-diagonal elements, but found the identity matrix to work best.

In order to compose a phrase representation \vec{p} from \vec{a} and \vec{n} , \mathbf{T} is applied to the adjective vector in a tensor dot product. The tensor dot product multiplies components of vector and tensor and sums along the third axis of the tensor:

$$\mathbf{X}_{i,j} = \sum_{k=1}^d a_k \cdot T_{i,j,k} \quad (6)$$

with d being the dimensionality of the word embeddings. Equation (6) results in a matrix \mathbf{X} that is multiplied with the noun vector in a second step using common matrix multiplication: $\vec{p} = \mathbf{X} \cdot \vec{n}$.

Note that the latter step corresponds to *functional application* of the adjective to the noun as rooted in compositional distributional semantics (Baroni et al., 2014a). The result is a phrase vector with the same dimensionality as adjective and noun. For initialization, we use an identity matrix for each second-order tensor along the third axis².

3.2.3 Training Method

The weights of the models in Section 3.2.2 are trained as part of a shallow neural network with no hidden layer. For each adjective-noun phrase and the corresponding ground truth attribute in the training dataset, the respective 300-dimensional vectors³ \vec{a} , \vec{n} and \vec{attr} are obtained by performing a look-up in the pre-trained word embeddings.

With \vec{a} and \vec{n} as its inputs, the neural network computes a phrase representation $\vec{p} \in \mathbb{R}^{300}$ at the output layer. The error of the computed phrase representation to the expected attribute representation \vec{attr} is computed using the *mean squared error* between the two vectors and is used as the training signal for the network parameters. Note that we do not train the embedding vectors along with the connection weights. While this could potentially benefit the results, we aim to explore whether generally trained word embeddings can be used to retrieve attribute meaning.

For our network architectures and computations, we use the deep learning library *keras* (Chollet, 2016). Training takes 10 iterations over the training data; weights are optimized using the stochastic optimization method *Adam* (Kingma and Ba, 2015). For the use of pre-trained word

²We found a random initialization of all entries to perform substantially worse.

³This is the number of dimensions in the pre-trained word embeddings from Mikolov et al. (2013b).

vectors (Mikolov et al., 2013b)⁴ in a Python environment, we rely on the *Gensim* library (Řehůřek and Sojka, 2010).

4 Attribute Selection Experiments

In this experiment, we evaluate the compositional models defined in Section 3.2 on the attribute selection task.

4.1 Data

We use the HeiPLAS data set (Hartung, 2015) which contains adjective-attribute-noun triples that were heuristically extracted from WordNet (Miller and Fellbaum, 1998) and manually filtered by linguistic curators. The data is separated into development and test set (comprising 869 and 729 triples, respectively, which correspond to a total of 254 target attributes). The target attributes are subdivided into various semantically homogeneous subsets, as shown in Table 1. Due to coverage issues in the pre-trained word2vec embeddings (Mikolov et al., 2013a), some adjectives and nouns from HeiPLAS cannot be projected into the embedding space⁵.

4.2 Experiment 1: Large-scale Attribute Selection

Experimental Procedure. Composition models as described in Section 3.2.2 are trained on all triples in HeiPLAS-Dev (following the procedure described in Section 3.2.3) and evaluated on HeiPLAS-Test. The word vector representations corresponding to the adjective and the noun in a test triple are composed into a phrase vector by applying the trained composition function. Using nearest neighbour search in \mathcal{S} as described in Section 3.2, all test attributes are ranked wrt. their similarity to the composed phrase vector. For evaluation, we use *precision-at-rank* to measure the number of times the correct attribute is ranked as most similar to the phrase vector or among the first five ranks (P@1 and P@5, respectively).

Baseline Semantic Spaces. We directly compare our approach against the results of two count-based distributional models, C-LDA and L-LDA (Hartung, 2015), on the same evaluation data. C-LDA and L-LDA induce distributional adjective

and noun vectors over attributes as dimensions of meaning, which are composed into phrase representations using pointwise vector multiplication. Using these models for comparison enables us to assess both the impact of different types of word representations (dense CBOW word embeddings vs. specifically tailored attribute-based distributional word vectors) and different approaches to compositionality (pre-defined vector mixture operations on attribute-specific word representations vs. trained composition functions for promoting generalized attribute meaning in word embeddings).

Results. Results of Experiment 1 are shown in Table 2. The upper part of the table contains the results based on word embeddings (comprising non-parametric, parametric, dilation and trainable composition models); the count-based C-LDA and L-LDA baselines are displayed below.

Focussing on the non-parametric models first, we find that relying on the adjective embedding as a surrogate of a composed representation already outperforms both count models by a wide margin. This indicates a clear advantage of CBOW embeddings over count-based representations for capturing attribute meaning at the word level. However, this holds only for adjectives; noun embeddings in isolation perform much worse.

This is confirmed by the dilation results: Dilating the noun representation into the direction of the adjective performs considerably better than vice versa, while there is no improvement beyond the non-compositional adjective baseline. These findings are in line with Hartung (2015) and Hartung and Frank (2011a) who also observed that adjective representations capture more of the compositional attribute semantics in adjective-noun phrases than noun representations do.

Considering the trained composition models, we find that weighting either the adjective or the noun in a full additive model substantially outperforms the respective non-compositional baseline. The overall best results are obtained by assigning trained weights to both the adjective and the noun embedding (P@1=0.56). This model also outperforms weighted vector addition⁶ using scalar weights by great margins.

⁴Available from <https://drive.google.com/file/d/0B7XkCwpI5KDYN1NUTt1SS21pQmM/edit?usp=sharing>

⁵This affects 54 triples in HeiPLAS-Dev and 44 triples in HeiPLAS-Test, which were removed from the evaluation.

⁶The weighted vector addition scores shown in Table 2 are based on optimized parameters as reported by Mitchell and Lapata (2010): $\alpha=0.88$ and $\beta=0.12$. By shifting the parameters further into the direction of the adjective (i.e., $\alpha=0.90$; $\beta=0.10$), P@1 slightly increases to 0.34.

Subset	Num. Attributes	Num. Train. Triples	Example Phrases
Core	10	72	<i>silvery hair</i> (COLOR), <i>huge wave</i> (SIZE), <i>longstanding conflict</i> (DURATION)
Selected	23	153	<i>sufficient food</i> (QUANTITY), <i>grave decision</i> (IMPORTANCE), <i>broad river</i> (WIDTH)
Measurable	65	261	<i>heavy load</i> (WEIGHT), <i>short hair</i> (LENGTH), <i>slow walker</i> (SPEED)
Property	73	300	<i>young people</i> (AGE), <i>high mountain</i> (HEIGHT), <i>straight line</i> (SHAPE)
All	254	869	<i>dry paint</i> (WETNESS), <i>scentless wisp</i> (SMELL), <i>vehement defense</i> (STRENGTH)

Table 1: Overview of subsets of attributes contained in HeiPLAS data, together with example phrases

Compositional Model		P@1	P@5	
predict models	Adjective	0.33	0.50	
	Noun	0.03	0.10	
	Vector Addition (\oplus)	0.24	0.45	
	Weighted Vector Addition	0.33	0.51	
	Vector Multiplication (\odot)	0.00	0.02	
	Adj. Dilation ($\lambda = 2$)	0.06	0.18	
	Noun Dilation ($\lambda = 2$)	0.33	0.51	
	Full Add. Weighted Noun	0.33	0.54	
	Full Add. Weighted Adjective	0.46	0.71	
	Full Add. Weighted Adj. and Noun	0.56	0.75	
	Trained Tensor Product (\otimes)	0.44	0.57	
	count	C-LDA (Hartung, 2015)	0.09	n/a
		L-LDA (Hartung, 2015)	0.16	n/a

Table 2: Results of Experiment 1; evaluation on all phrases from HeiPLAS-Test

In comparison to the best full additive model, the tensor product underperforms by more than 10 points in P@1 and also falls short of weighting only the adjective. This is in line with a general preference of word embeddings for additive models (Mikolov et al., 2013a), which is also confirmed by the non-parametric composition functions. On the other hand, we conjecture that the relatively small size of the training set used here is not sufficient for optimally tuning the 300^3 parameters in the learned tensor.

4.3 Experiment 2: Generalization Power

In this experiment, we are interested in assessing the generalization power of the best-performing composition function as trained in Experiment 1. More precisely, we investigate the hypothesis that a full additive model captures a generalized compositional process in the semantics of attribute-denoting adjective-noun phrases rather than the lexical meaning of individual attributes (cf. Bride et al. (2015)).

We evaluate this hypothesis wrt. (i) the fit of the composition function to different subsets of testing

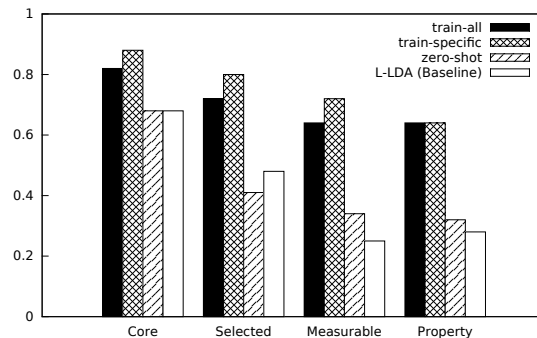


Figure 1: Attribute selection performance of the full additive model after training on all attributes, specific subsets, and in zero-shot learning

attributes, and (ii) its predictive capacity in a zero-shot learning scenario.

Subsets of Testing Attributes. First, we compare the fit of the composition function that has been trained on all attributes (cf. Experiment 1) on the different subsets of attributes in HeiPLAS-Test, as displayed in Table 1.

The results of this experiment are shown in Figure 1. As can be seen from the solid bars in the plot, the attribute selection performance on individual subsets is considerably stronger than on the entire inventory, ranging from P@1=0.82 on the Core subset to P@1=0.64 on the Property and Measurable subsets (compared to P@1=0.56 on all attributes; cf. Table 2). The cross-hatched bars in the figure indicate the relative differences that result from re-training a composition function on the specific subset of interest. The improvements are consistently small (max. +0.08 on the Selected and Measurable subsets); in case of the Property subset, there is no difference at all.

Zero-Shot Learning. As defined by Palatucci et al. (2009), zero-shot learning is the task of learning a classifier for predicting novel class labels un-

seen during training. In order to assess the selection performance of our model in a zero-shot setting, we create four zero-shot training sets by removing from HeiPLAS-Train all attributes that are contained in each of the subsets described in Table 1, respectively. The corresponding subset from HeiPLAS-Test is used for evaluation afterwards.

The zero-shot results are shown by the diagonally hatched bars in Fig. 1. We find that Core attributes, without being seen during training, can be predicted at a performance of $P@1=0.68$. On larger subsets, zero-shot performance decreases (down to $P@1=0.32$ on Property attributes). Yet, we consider these results very decent overall, given that they are largely comparable or even superior (except for the Selected subset) to the best scores of the distributional L-LDA model (Hartung, 2015) as shown by the plain bars in Fig. 1.

Even though benefits from attribute-specific training cannot be denied, we find that the trained compositionality function is largely capable of generalizing over individual target attributes.

4.4 Discussion

Our experiments on attribute selection show that CBOW word embeddings can be effectively harnessed for carving out attribute meaning from adjective-noun phrases. Observed improvements over the previous state-of-the-art are due to the type of word representation as such (dense neural embeddings vs. distributional count models) as well as a learned compositionality function based on a full additive model capitalizing on weight matrices for balancing the contributions of adjectives and nouns. Moreover, we were able to show that the compositionality function captures a generalized compositional process in the semantics of attribute-denoting adjective-noun phrases rather than the lexical meaning of individual attributes. Therefore, the proposed approach (i) poses an interesting alternative to previous distributional models which explicitly encode attribute meaning in word vectors and rely on vector mixture operations in order to compose them into attribute-based phrase representations, and (ii) bears the potential of being used as a generalized attribute extraction model on various domains of applications that demand for different attribute inventories.

5 Similarity Prediction Experiments

In this experiment, we assess the scalability of the previously trained composition models to different tasks by applying them to the prediction of semantic similarity in pairs of adjective-noun phrases.

5.1 Data

Our experiments are based on the adjective-noun section of the evaluation data set released by Mitchell and Lapata (2010). It consists of 108 pairs of adjective-noun phrases that were rated for similarity on a 7-point scale⁷ by 54 human judges. In total, the data set comprises 1944 data points.

5.2 Experiment 3: Predicting Adjective-Noun Phrase Similarity

Experimental Procedure. For a given pair of adjective-noun phrases, we compute two phrase representations using word embeddings as word representations and compositionality functions trained on the HeiPLAS-Core subset, which achieved the best attribute selection results in Experiments 1 and 2. In the next step, we compute the cosine similarity between these two phrase representations. We correlate the results with human similarity ratings using Spearman’s ρ and compare the resulting correlation scores to the reported results of Mitchell and Lapata (2010).

Baseline Models. We compare our models against the following approaches from the literature which were evaluated on the same data set: C-LDA (Hartung and Frank, 2011a), M&L-BoW and M&L-Topic (both by Mitchell and Lapata (2010)). All baseline models are count-based distributional models which differ in their underlying representation of word meaning: M&L-BoW relies on bag-of-words context windows, M&L-Topic and C-LDA use topics and attribute nouns as dimensions of meaning, respectively.

Results. As shown in Table 3, the best correlation scores between human similarity judgments and model predictions are achieved by our model that is built upon word embeddings and a trained full additive composition function based on weighting adjective and noun vectors ($\rho=0.50$). This model outperforms all distributional baseline models using vector mixtures as composition functions.

⁷A score of 1 expresses low similarity between phrases, 7 indicates high similarity.

Underlying Word Representation	\odot	\oplus	Weighted Addition	Full Additive
word2vec	0.36	0.48	0.42	0.50
M&L-BoW	0.46	0.36	0.44	n/a
M&L-Topic	0.25	0.37	0.38	n/a
C-LDA	0.28	0.19	n/a	n/a

Table 3: Results of Experiment 3 (Spearman’s ρ between human judgments and model predictions)

With respect to weighted addition, all results reported in Table 3 are based on the weighting parameters ($\alpha=0.88$; $\beta=0.12$) that have been found as optimal by Mitchell and Lapata (2010). Based on a grid search, we find $\alpha=0.60$ and $\beta=0.40$ to be the best weighting parameters on our data. In this setting, the performance of the weighted vector addition model on word2vec embeddings can be increased to $\rho=0.47$, which is still slightly below unweighted vector addition on embeddings ($\rho=0.48$). Apparently, scalar weights in pointwise vector addition are quite sensitive to the underlying word representation. In the particular case of using word embeddings for similarity prediction, the contribution of the noun to the compositional semantics of the phrase seems to be relatively stronger than in the attribute selection task (cf. Experiment 1).

In total, these results indicate that compositionality functions optimized on the task of attribute selection can be effectively transferred to similarity prediction. This suggests that attribute meaning might be a prominent source of similarity in adjective-noun phrases, which will be subject to a closer investigation in the next experiment.

5.3 Experiment 4: Interpreting the Source of Similarity

Research in distributional semantics tends to focus on the *degree* of similarity between words or phrases, while the *source* of similarity is largely neglected (cf. Hartung (2015)). In this experiment, we hypothesize that attribute meaning provides a plausible explanation for the observed degree of similarity in phrase pairs from the M&L data set.

Experimental Procedure. For a given phrase pair, we compute the top-5 most similar attributes for each phrase in terms of their nearest neighbours in \mathcal{S} (cf. Section 3.2). Then, both phrases

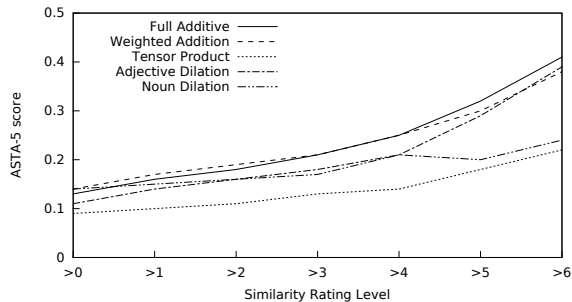


Figure 2: ASTA-5 scores over different levels of human similarity ratings (cf. Experiment 4)

are compared wrt. the proportion of shared attributes within these top-5 predictions. Averaging this score over all phrase pairs which were assigned a particular similarity rating by the human judges yields an *Average Shared Top-5 Attributes* (ASTA-5) score for this similarity level.

Results. Figure 2 plots ASTA-5 scores at different levels of human similarity ratings. We observe a general trend across all compositionality functions investigated: The higher the rating cutoff, the higher the number of shared attributes. Thus, with increasing similarity between two phrases (according to human ratings), the proportion of shared attributes in their compositional semantics tends to increase as well. Moreover, for highly similar pairs (rating cutoff >5), the full additive vector addition model yields the highest ASTA-5 scores.

Beyond this quantitative analysis, two of the authors manually investigated the shared attributes in 38 high-similarity phrase pairs (rating cutoff >4) as predicted by the weighted vector addition model wrt. their potential as plausible sources of similarity. We find that in 28 phrase pairs (73.6%), the predicted attribute is considered a plausible source of similarity, in eight others (26.4%), the predicted attribute does not explain the high similarity. The agreement between the annotators in terms of Fleiss’ Kappa amounts to $\kappa = 0.62$.

5.4 Discussion

Our results show that a full additive compositional model trained to target attribute meaning improves performance on similarity prediction. This supports the interpretation that attributes are (at least)

a partial source of similarity between adjective-noun phrases. In fact, this has been corroborated by a preliminary manual investigation of shared attributes between high-similarity phrases. However, there is also evidence for several cases in which attribute meaning falls short of explaining high phrase similarity. This holds for phrases involving abstract concepts, for instance (cf. Hartung (2015), Borghi and Binkofski (2014)).

Nevertheless, we consider it a strength of our model that it is capable of providing plausible explanations in cases where attribute meaning is the most prominent source of similarity.

6 Conclusions

We have presented a model of attribute meaning in adjective-noun phrases that capitalizes on CBOW word embeddings. In our experiments, the model proves remarkably versatile as it advances the state-of-the-art in the two tasks of attribute selection and phrase similarity prediction. In the latter task, the property of being fully interpretable wrt. attributes as the potential source of similarities became apparent as an additional asset rendering the model potentially interoperable with knowledge representation formalisms and resources.

Improvements over previous distributional models can be traced back to two major sources: First, CBOW word embeddings work surprisingly well at the word level for capturing attribute meaning in adjectives (not for nouns, though). Future work should investigate whether further improvements can be obtained from more adjective-specific word embeddings that are trained on symmetric coordination patterns (Schwartz et al., 2016). Second, a learned compositionality function is effective at promoting attribute meaning in composed phrase representations. Best performances across both tasks are achieved by a full additive model with distinct weight matrices for the adjective and noun constituent. A trained tensor product that comes closer to the linguistic notion of functional application also performs well beyond the previous state-of-the-art, while falling short of the additive model. Apparently, more training data is needed to exhaust the full potential of the tensor product. Alternatively, tensor decomposition techniques along the lines of Shah et al. (2015) may be a possible way of coping with the large parameter

space of the tensor approach.

Moreover, the learned compositionality function turns out to generalize well over individual attributes, which we consider a very promising result wrt. the suitability of the model in various NLP tasks such as aspect-based sentiment analysis. In future work, we are going to extend the present model to consider broader linguistic contexts and more varied syntactic configurations.

Acknowledgments

We gratefully acknowledge feedback and comments by the anonymous EACL reviewers, which considerably helped to improve the paper. This work was supported by the Cluster of Excellence *Cognitive Interaction Technology* 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG), and by the German Federal Ministry of Education and Research (BMBF) in the *KogniHome* project.

References

- Abdulrahman Almuhaieb. 2006. *Attributes in lexical acquisition*. Ph.D. thesis, University of Essex.
- Omid Bakshshandeh and James F. Allen. 2015. From Adjective Glosses to Attribute Concepts: Learning Different Aspects That an Adjective Can Describe. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*, pages 23–33, London, UK.
- Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.
- Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014a. Frege in Space: A Program for Compositional Distributional Semantics. *Linguistic Issues in Language Technology*, 9:241–346.
- Marco Baroni, Georgiana Dinu, and Germà Kruszewski. 2014b. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In Kristina Toutanova and Hua Wu, editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.
- Lawrence W. Barsalou. 1992. Frames, Concepts and Conceptual Fields. In A. Lehrer and E.F. Kittay, editors, *Frames, Fields and Contrasts*, pages 21–74. Lawrence Erlbaum Associates, Hillsdale, NJ.

- Anna M. Borghi and Ferdinand Binkofski. 2014. *Words as Social Tools: An Embodied View on Abstract Concepts*. Springer Briefs in Cognition. Springer.
- Antoine Bride, Tim Van de Cruys, and Nicholas Asher. 2015. A Generalisation of Lexical Functions for Composition in Distributional Semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 281–291, Beijing, China, July. Association for Computational Linguistics.
- François Chollet. 2016. keras. <https://github.com/fchollet/keras>.
- Philipp Cimiano. 2006. *Ontology Learning and Population from Text. Algorithms, Evaluation and Applications*. Springer.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- Silvio Cordeiro, Carlos Ramisch, Marco Idiart, and Aline Villavicencio. 2016. Predicting the Compositionality of Nominal Compounds: Giving Word Embeddings a Hard Time. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1986–1997, Berlin, Germany, August. Association for Computational Linguistics.
- Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.
- Katrin Erk. 2016. What do you know about an alligator when you know the company it keeps? *Semantics & Pragmatics*, 9:1–63.
- Gottlob Frege. 1892. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50.
- Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.
- Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2014. Concrete Sentence Spaces for Compositional Distributional Models of Meaning. In Harry Bunt, Johan Bos, and Stephen Pulman, editors, *Computing Meaning*, volume 4, pages 71–86. Springer.
- Nicola Guarino. 1992. Concepts, Attributes and Arbitrary Relations. *Data & Knowledge Engineering*, 8:249–261.
- Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In Roberto Basili and Marco Pennacchiotti, editors, *Proceedings of the 2010 Workshop on Geometrical Models of Natural Language Semantics*, pages 33–37, Uppsala, Sweden, July. Association for Computational Linguistics.
- Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.
- Matthias Hartung and Anette Frank. 2010. A Structured Vector Space Model for Hidden Attribute Meaning in Adjective-Noun Phrases. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, Beijing, China, pages 430–438.
- Matthias Hartung and Anette Frank. 2011a. Assessing interpretable, attribute-related meaning representations for adjective-noun phrases in a similarity prediction task. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 52–61, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Matthias Hartung and Anette Frank. 2011b. Exploring Supervised LDA Models for Assigning Attributes to Adjective-Noun Phrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 540–551, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Matthias Hartung. 2015. *Distributional Semantic Models of Attribute Meaning in Adjectives and Nouns*. Ph.D. thesis, Heidelberg University.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.
- Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.
- Sebastian Löbner. 2013. *Understanding Semantics*. Routledge, 2nd edition.
- Christopher D. Manning. 2015. Computational Linguistics and Deep Learning. *Computational Linguistics*, 41:701–707.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop*.
- George Miller and Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.

- Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.
- Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In *Proceedings of NIPS*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.
- Wiebke Petersen and Oliver Hellwig. 2016. Exploring the value space of attributes: Unsupervised bidirectional clustering of adjectives in German. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2839–2848, Osaka, Japan, December. The COLING 2016 Organizing Committee.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In René Witte, Hamish Cunningham, Jon Patrick, Elena Beisswanger, Ekaterina Buyko, Udo Hahn, Karin Verspoor, and Anni R. Coden, editors, *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA.
- Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 977–983, Denver, Colorado, May–June. Association for Computational Linguistics.
- Roy Schwartz, Roi Reichart, and Ari Rappoport. 2016. Symmetric patterns and coordinations: Fast and enhanced representations of verbs and adjectives. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 499–505, San Diego, California, June. Association for Computational Linguistics.
- Parikshit Shah, Nikhil Rao, and Gongguo Tang. 2015. Sparse and low-rank tensor decomposition. In *Proceedings of NIPS*.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Niket Tandon, Gerard de Melo, Fabian Suchanek, and Gerhard Weikum. 2014. WebChild: Harvesting and Organizing Commonsense Knowledge from the Web. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 523–532, New York, NY, USA. ACM.
- Dominic Widdows. 2008. Semantic Vector Products: Some Initial Investigations. In *Proceedings of the 2nd Conference on Quantum Interaction*, Oxford, UK.

Hypernyms under Siege: Linguistically-motivated Artillery for Hypernymy Detection

Vered Shwartz¹, Enrico Santus^{2,3} and Dominik Schlechtweg⁴

¹Bar-Ilan University, Ramat-Gan, Israel

²Singapore University of Technology and Design, Singapore

³The Hong Kong Polytechnic University, Hong Kong

⁴University of Stuttgart, Stuttgart, Germany

{vered1986, esantus}@gmail.com, dominik.schlechtweg@gmx.de

Abstract

The fundamental role of hypernymy in NLP has motivated the development of many methods for the automatic identification of this relation, most of which rely on word distribution. We investigate an extensive number of such unsupervised measures, using several distributional semantic models that differ by context type and feature weighting. We analyze the performance of the different methods based on their linguistic motivation. Comparison to the state-of-the-art supervised methods shows that while supervised methods generally outperform the unsupervised ones, the former are sensitive to the distribution of training instances, hurting their reliability. Being based on general linguistic hypotheses and independent from training data, unsupervised measures are more robust, and therefore are still useful artillery for hypernymy detection.

1 Introduction

In the last two decades, the NLP community has invested a consistent effort in developing automated methods to recognize hypernymy. Such effort is motivated by the role this semantic relation plays in a large number of tasks, such as taxonomy creation (Snow et al., 2006; Navigli et al., 2011) and recognizing textual entailment (Dagan et al., 2013). The task has appeared to be, however, a challenging one, and the numerous approaches proposed to tackle it have often shown limitations.

Early corpus-based methods have exploited patterns that may indicate hypernymy (e.g. “*animals* such as *dogs*”) (Hearst, 1992; Snow et al., 2005), but the recall limitation of this approach, requiring both words to co-occur in a sentence, motivated the development of methods that rely on

adaptations of the *distributional hypothesis* (Harris, 1954).

The first distributional approaches were unsupervised, assigning a score for each (x, y) word-pair, which is expected to be higher for hypernym pairs than for negative instances. Evaluation is performed using ranking metrics inherited from information retrieval, such as Average Precision (AP) and Mean Average Precision (MAP). Each measure exploits a certain linguistic hypothesis such as the *distributional inclusion hypothesis* (Weeds and Weir, 2003; Kotlerman et al., 2010) and the *distributional informativeness hypothesis* (Santus et al., 2014; Rimell, 2014).

In the last couple of years, the focus of the research community shifted to supervised distributional methods, in which each (x, y) word-pair is represented by a combination of x and y 's word vectors (e.g. concatenation or difference), and a classifier is trained on these resulting vectors to predict hypernymy (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). While the original methods were based on count-based vectors, in recent years they have been used with word embeddings (Mikolov et al., 2013; Pennington et al., 2014), and have gained popularity thanks to their ease of use and their high performance on several common datasets. However, there have been doubts on whether they can actually learn to recognize hypernymy (Levy et al., 2015b).

Additional recent hypernymy detection methods include a multimodal perspective (Kiela et al., 2015), a supervised method using unsupervised measure scores as features (Santus et al., 2016a), and a neural method integrating path-based and distributional information (Shwartz et al., 2016).

In this paper we perform an extensive evaluation of various unsupervised distributional measures for hypernymy detection, using several distributional semantic models that differ by context type and feature weighting. Some measure vari-

ants and context-types are tested for the first time.¹

We demonstrate that since each of these measures captures a different aspect of the hypernymy relation, there is no single measure that consistently performs well in discriminating hypernymy from different semantic relations. We analyze the performance of the measures in different settings and suggest a principled way to select the suitable measure, context type and feature weighting according to the task setting, yielding consistent performance across datasets.

We also compare the unsupervised measures to the state-of-the-art supervised methods. We show that supervised methods outperform the unsupervised ones, while also being more efficient, computed on top of low-dimensional vectors. At the same time, however, our analysis reassesses previous findings suggesting that supervised methods do not actually learn the relation between the words, but only characteristics of a single word in the pair (Levy et al., 2015b). Moreover, since the features in embedding-based classifiers are latent, it is difficult to tell what the classifier has learned. We demonstrate that unsupervised methods, on the other hand, do account for the relation between words in a pair, and are easily interpretable, being based on general linguistic hypotheses.

2 Distributional Semantic Spaces

We created multiple distributional semantic spaces that differ in their context type and feature weighting. As an underlying corpus we used a concatenation of the following two corpora: `ukWaC` (Ferraresi, 2007), a 2-billion word corpus constructed by crawling the `.uk` domain, and `WaCkypedia_EN` (Baroni et al., 2009), a 2009 dump of the English Wikipedia. Both corpora include POS, lemma and dependency parse annotations. Our vocabulary (of target and context words) includes only nouns, verbs and adjectives that occurred at least 100 times in the corpus.

Context Type We use several context types:

- **Window-based contexts:** the contexts of a target word w_i are the words surrounding it in a k -sized window: $w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k}$. If the context-type is directional, words occurring before and after w_i are marked differently, i.e.: $w_{i-k}/l, \dots, w_{i-1}/l, w_{i+1}/r, \dots, w_{i+k}/r$.

¹Our code and data are available at:
<https://github.com/vered1986/UnsupervisedHypernymy>

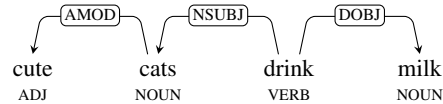


Figure 1: An example dependency tree of the sentence *cute cats drink milk*, with the target word *cats*. The dependency-based contexts are *drink-v:nsbj* and *cute-a:amod*⁻¹. The joint-dependency context is *drink-v#milk-n*. Differently from Chersoni et al. (2016), we exclude the dependency tags to mitigate the sparsity of contexts.

Out-of-vocabulary words are filtered out before applying the window. We experimented with window sizes 2 and 5, directional and indirectional (`win2`, `win2d`, `win5`, `win5d`).

- **Dependency-based contexts:** rather than adjacent words in a window, we consider neighbors in a dependency parse tree (Padó and Lapata, 2007; Baroni and Lenci, 2010). The contexts of a target word w_i are its parent and daughter nodes in the dependency tree (`dep`). We also experimented with a joint dependency context inspired by Chersoni et al. (2016), in which the contexts of a target word are the parent-sister pairs in the dependency tree (`joint`). See Figure 1 for an illustration.

Feature Weighting Each distributional semantic space is spanned by a matrix M in which each row corresponds to a target word while each column corresponds to a context. The value of each cell $M_{i,j}$ represents the association between the target word w_i and the context c_j . We experimented with two feature weightings:

- **Frequency** - raw frequency (no weighting): $M_{i,j}$ is the number of co-occurrences of w_i and c_j in the corpus.
- **Positive PMI (PPMI)** - pointwise mutual information (PMI) (Church and Hanks, 1990) is defined as the log ratio between the joint probability of w and c and the product of their marginal probabilities: $PMI(w, c) = \log \frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}(c)}$, where $\hat{P}(w)$, $\hat{P}(c)$, and $\hat{P}(w, c)$ are estimated by the relative frequencies of a word w , a context c and a word-context pair (w, c) , respectively. To handle unseen pairs (w, c) , yielding $PMI(w, c) = \log(0) = -\infty$, PPMI (Bullinaria and Levy, 2007) assigns zero to negative PMI scores: $PPMI(w, c) = \max(PMI(w, c), 0)$.

In addition, one of the measures we used (San-tus et al., 2014) required a third feature weighting:

- **Positive LMI (PLMI)** - positive local mutual information (PLMI) (Evert, 2005; Evert, 2008). PPMI was found to have a bias towards rare events. PLMI simply balances PPMI by multiplying it by the co-occurrence frequency of w and c : $PLMI(w, c) = freq(w, c) \cdot PPMI(w, c)$.

3 Unsupervised Hypernymy Detection Measures

We experiment with a large number of unsupervised measures proposed in the literature for distributional hypernymy detection, with some new variants. In the following section, \vec{v}_x and \vec{v}_y denote x and y 's word vectors (rows in the matrix M). We consider the scores as measuring to what extent y is a hypernym of x ($x \rightarrow y$).

3.1 Similarity Measures

Following the *distributional hypothesis* (Harris, 1954), similar words share many contexts, thus have a high similarity score. Although the hypernymy relation is asymmetric, similarity is one of its properties (Santus et al., 2014).

- **Cosine Similarity** (Salton and McGill, 1986) A symmetric similarity measure:

$$\cos(x, y) = \frac{\vec{v}_x \cdot \vec{v}_y}{\|\vec{v}_x\| \cdot \|\vec{v}_y\|}$$

- **Lin Similarity** (Lin, 1998) A symmetric similarity measure that quantifies the ratio of shared contexts to the contexts of each word:

$$Lin(x, y) = \frac{\sum_{c \in \vec{v}_x \cap \vec{v}_y} [\vec{v}_x[c] + \vec{v}_y[c]]}{\sum_{c \in \vec{v}_x} \vec{v}_x[c] + \sum_{c \in \vec{v}_y} \vec{v}_y[c]}$$

- **APSyn** (Santus et al., 2016b) A symmetric measure that computes the extent of intersection among the N most related contexts of two words, weighted according to the rank of the shared contexts (with N as a hyper-parameter):

$$APSyn(x, y) = \sum_{c \in N(\vec{v}_x) \cap N(\vec{v}_y)} \frac{1}{\frac{rank_x(c) + rank_y(c)}{2}}$$

3.2 Inclusion Measures

According to the *distributional inclusion hypothesis*, the prominent contexts of a hyponym (x) are expected to be included in those of its hypernym (y).

- **Weeds Precision** (Weeds and Weir, 2003) A directional precision-based similarity measure. This measure quantifies the weighted inclusion of x 's contexts by y 's contexts:

$$WeedsPrec(x \rightarrow y) = \frac{\sum_{c \in \vec{v}_x \cap \vec{v}_y} \vec{v}_x[c]}{\sum_{c \in \vec{v}_x} \vec{v}_x[c]}$$

- **cosWeeds** (Lenci and Benotto, 2012) Geometric mean of cosine similarity and Weeds precision:

$$\cosWeeds(x \rightarrow y) = \sqrt{\cos(x, y) \cdot WeedsPrec(x \rightarrow y)}$$

- **ClarkeDE** (Clarke, 2009) Computes degree of inclusion, by quantifying weighted coverage of the hyponym's contexts by those of the hypernym:

$$CDE(x \rightarrow y) = \frac{\sum_{c \in \vec{v}_x \cap \vec{v}_y} \min(\vec{v}_x[c], \vec{v}_y[c])}{\sum_{c \in \vec{v}_x} \vec{v}_x[c]}$$

- **balAPinc** (Kotlerman et al., 2010) Balanced average precision inclusion.

$$APinc(x \rightarrow y) = \frac{\sum_{r=1}^{N_y} [P(r) \cdot rel(c_r)]}{N_y}$$

is an adaptation of the average precision measure from information retrieval for the inclusion hypothesis. N_y is the number of non-zero contexts of y and $P(r)$ is the precision at rank r , defined as the ratio of shared contexts with y among the top r contexts of x . $rel(c)$ is the relevance of a context c , set to 0 if c is not a context of y , and to $1 - \frac{rank_y(c)}{N_y + 1}$ otherwise, where $rank_y(c)$ is the rank of the context c in y 's sorted vector. Finally,

$$balAPinc(x \rightarrow y) = \sqrt{Lin(x, y) \cdot APinc(x \rightarrow y)}$$

is the geometric mean of APinc and Lin similarity.

- **invCL** (Lenci and Benotto, 2012) Measures both distributional inclusion of x in y and distributional non-inclusion of y in x :

$$invCL(x \rightarrow y) = \sqrt{CDE(x \rightarrow y) \cdot (1 - CDE(y \rightarrow x))}$$

3.3 Informativeness Measures

According to the *distributional informativeness hypothesis*, hypernyms tend to be less informative than hyponyms, as they are likely to occur in more general contexts than their hyponyms.

- **SLQS** (Santus et al., 2014)

$$SLQS(x \rightarrow y) = 1 - \frac{E_x}{E_y}$$

The informativeness of a word x is evaluated as the median entropy of its top N contexts: $E_x = \text{median}_{i=1}^N (H(c_i))$, where $H(c)$ is the entropy of context c .

- **SLQS Sub** A new variant of SLQS based on the assumption that if y is judged to be a hypernym of x to a certain extent, then x should be judged to be a hyponym of y to the same extent (which is not the case for regular SLQS). This is achieved by subtraction:

$$SLQS_{sub}(x \rightarrow y) = E_y - E_x$$

It is weakly symmetric in the sense that $SLQS_{sub}(x \rightarrow y) = -SLQS_{sub}(y \rightarrow x)$.

SLQS and SLQS Sub have 3 hyper-parameters: i) the number of contexts N ; ii) whether to use median or average entropy among the top N contexts; and iii) the feature weighting used to sort the contexts by relevance (i.e., PPMI or PLMI).

- **SLQS Row** Differently from SLQS, SLQS Row computes the entropy of the target rather than the average/median entropy of the contexts, as an alternative way to compute the generality of a word.² In addition, parallel to SLQS we tested SLQS Row with subtraction, **SLQS Row Sub**.
- **RCTC** (Rimell, 2014) Ratio of change in topic coherence:

$$RCTC(x \rightarrow y) = \frac{TC(t_x)/TC(t_{x \setminus y})}{TC(t_y)/TC(t_{y \setminus x})}$$

where t_x are the top N contexts of x , considered as x 's *topic*, and $t_{x \setminus y}$ are the top N contexts of x which are not contexts of y . $TC(A)$ is the topic coherence of a set of words A , defined as the median pairwise PMI scores between words in A . N is a hyper-parameter. The measure is based on the assumptions that excluding y 's contexts from x 's increases the coherence of the topic, while excluding x 's contexts from y 's decreases the coherence of the topic. We include this measure under the informativeness inclusion, as it is based on a similar hypothesis.

3.4 Reversed Inclusion Measures

These measures are motivated by the fact that, even though—being more general—hypernyms are expected to occur in a larger set of contexts, sentences like “the *vertebrate* barks” or “the *mammal* arrested the thieves” are not common, since hyponyms are more specialized and are hence more appropriate in such contexts. On the other

²In our preliminary experiments, we noticed that the entropies of the targets and those of the contexts are not highly correlated, yielding a Spearman’s correlation of up to 0.448 for window based spaces, and up to 0.097 for the dependency-based ones ($p < 0.01$).

dataset	relations	#instances	size
BLESS	hypernym	1,337	26,554
	meronym	2,943	
	coordination	3,565	
	event	3,824	
	attribute	2,731	
	random-n	6,702	
	random-v	3,265	
EVALution	hypernym	2,187	13,465 ³
	hypernym	3,637	
	meronym	1,819	
	attribute	2,965	
	synonym	1,888	
Lenci/Benotto	antonym	3,156	5,010
	hypernym	1,933	
	synonym	1,311	
Weeds	antonym	1,766	2,928
	hypernym	1,469	
	coordination	1,459	

Table 1: The semantic relations, number of instances in each relation, and size of each dataset.

hand, hyponyms are likely to occur in broad contexts (e.g. *eat*, *live*), where hypernyms are also appropriate. In this sense, we can define the *reversed inclusion hypothesis*: “hypernym’s contexts are likely to be included in the hyponym’s contexts”. The following variants are tested for the first time.

- **Reversed Weeds**

$$RevWeeds(x \rightarrow y) = Weeds(y \rightarrow x)$$

- **Reversed ClarkeDE**

$$RevCDE(x \rightarrow y) = CDE(y \rightarrow x)$$

4 Datasets

We use four common semantic relation datasets: BLESS (Baroni and Lenci, 2011), EVALution (Santus et al., 2015), Lenci/Benotto (Benotto, 2015), and Weeds (Weeds et al., 2014). The datasets were constructed either using knowledge resources (e.g. WordNet, Wikipedia), crowd-sourcing or both. The semantic relations and the size of each dataset are detailed in Table 1.

In our distributional semantic spaces, a target word is represented by the word and its POS tag. While BLESS and Lenci/Benotto contain this information, we needed to add POS tags to the other datasets. For each pair (x, y) , we considered 3 pairs $(x-p, y-p)$ for $p \in \{noun, adjective, verb\}$, and added the respective pair to the dataset only if the words were present in the corpus.⁴

³We removed the *entailment* relation, which had too few instances, and conflated relations to coarse-grained relations (e.g. *HasProperty* and *HasA* into *attribute*).

⁴Lenci/Benotto includes pairs to which more than one relation is assigned, e.g. when x or y are polysemous, and re-

dataset	hyper vs. relation	measure	context type	feature weighting	hyper-parameters	AP@100	AP@All
EVALution	all other relations	invCL	joint	freq	-	0.661	0.353
	meronym	APSyn	joint	freq	$N=500$	0.883	0.675
	attribute	APSyn	joint	freq	$N=500$	0.88	0.651
	antonym	SLQS_row	joint	freq	-	0.74	0.54
			joint	ppmi		0.74	0.55
			joint	plmi		0.74	0.537
	synonym	SLQS_row	joint	freq	-	0.83	0.647
			joint	ppmi		0.83	0.657
			joint	plmi		0.83	0.645
BLESS	all other relations	invCL	win5	freq	-	0.54	0.051
	meronym	SLQS _{sub}	win5d	freq	$N=100, median, plmi$	1.0	0.76
		SLQS	win5d	freq	$N=100, median, plmi$	1.0	0.758
	coord	SLQS _{sub}	joint	freq	$N=50, average, plmi$	0.995	0.537
	attribute	SLQS _{sub}	dep	plmi	$N=70, average, plmi$	1.0	0.74
		cosine	joint	freq	-	1.0	0.622
	event	APSyn	dep	freq	$N=1000$	1.0	0.779
Lenci/ Benotto	all other relations	APSyn	joint	freq	$N=1000$	0.617	0.382
	antonym	APSyn	dep	freq	$N=1000$	0.861	0.624
	synonym	SLQS_row _{sub}	joint	ppmi	-	0.948	0.725
Weeds	all other relations	clarkeDE	win5d	freq	-	0.911	0.441
	coord	clarkeDE	win5d	freq	-	0.911	0.441

Table 2: Best performing unsupervised measures on each dataset in terms of Average Precision (AP) at $k = 100$, for hypernym vs. all other relations and vs. each single relation. AP for $k = all$ is also reported for completeness. We excluded the experiments of hypernym vs. random-(n, v, j) for brevity; most of the similarity and some of the inclusion measures achieve $AP@100 = 1.0$ in these experiments.

We split each dataset randomly to 90% test and 10% validation. The validation sets are used to tune the hyper-parameters of several measures: SLQS (Sub), APSyn and RCTC.

5 Experiments

5.1 Comparing Unsupervised Measures

In order to evaluate the unsupervised measures described in Section 3, we compute the measure scores for each (x, y) pair in each dataset. We first measure the method’s ability to discriminate hypernymy from all other relations in the dataset, i.e. by considering hypernyms as positive instances, and other word pairs as negative instances. In addition, we measure the method’s ability to discriminate hypernymy from every other relation in the dataset by considering *one* relation at a time. For a relation \mathcal{R} we consider only (x, y) pairs that are annotated as either hypernyms (positive instances) or \mathcal{R} (negative instances). We rank the pairs according to the measure score and compute average precision (AP) at $k = 100$ and $k = all$.⁵

lated differently in each sense. We consider y as a hypernym of x if hypernymy holds in some of the words’ senses. Therefore, when a pair is assigned both hypernymy and another relation, we only keep it as hypernymy.

⁵We tried several cut-offs and chose the one that seemed to be more informative in distinguishing between the unsupervised measures.

Table 2 reports the best performing measure(s), with respect to $AP@100$, for each relation in each dataset. The first observation is that there is no single combination of measure, context type and feature weighting that performs best in discriminating hypernymy from all other relations. In order to better understand the results, we focus on the second type of evaluation, in which we discriminate hypernyms from each other relation.

The results show preference to the syntactic context-types (`dep` and `joint`), which might be explained by the fact that these contexts are richer (as they contain both proximity and syntactic information) and therefore more discriminative. In feature weighting there is no consistency, but interestingly, raw frequency appears to be successful in hypernymy detection, contrary to previously reported results for word similarity tasks, where PPMI was shown to outperform it (Bullinaria and Levy, 2007; Levy et al., 2015a).

The new SLQS variants are on top of the list in many settings. In particular they perform well in discriminating hypernyms from symmetric relations (antonymy, synonymy, coordination).

The measures based on the *reversed inclusion hypothesis* performed inconsistently, achieving perfect score in the discrimination of hypernyms from unrelated words, and performing well

relation	measure	context type	feature weighting
meronym	cosWeeds	dep	ppmi
	Weeds	dep / joint	ppmi
	ClarkeDE	dep / joint	ppmi / freq
attribute	APSyn	joint	freq
	cosine	joint	freq
	Lin	dep	ppmi
	cosine	dep	ppmi
antonym	SLQS	-	-
	SLQS_row	joint	(freq/ppmi/plmi)
synonym	SLQS_row/SLQS_row_sub	dep	ppmi
	invCL	win2/5/5d	freq
coordination		-	

Table 3: Intersection of datasets’ top-performing measures when discriminating between hypernymy and each other relation.

in few other cases, always in combination with syntactic contexts.

Finally, the results show that there is no single combination of measure and parameters that performs consistently well for all datasets and classification tasks. In the following section we analyze the best combination of measure, context type and feature weighting to distinguish hypernymy from any other relation.

5.2 Best Measure Per Classification Task

We considered all relations that occurred in two datasets. For such relation, for each dataset, we ranked the measures by their AP@100 score, selecting those with score ≥ 0.8 .⁶ Table 3 displays the intersection of the datasets’ best measures.

Hypernym vs. Meronym The inclusion hypothesis seems to be most effective in discriminating between hypernyms and meronyms under syntactic contexts. We conjecture that the window-based contexts are less effective since they capture topical context words, that might be shared also among holonyms and their meronyms (e.g. *car* will occur with many of the neighbors of *wheel*). However, since meronyms and holonyms often have different functions, their functional contexts, which are expressed in the syntactic context-types, are less shared. This is where they mostly differ from hyponym-hypernym pairs, which are of the same function (e.g. *cat* is a type of *animal*).

Table 2 shows that SLQS performs well in this task on BLESS. This is contrary to previous findings that suggested that SLQS is weak in discriminating between hypernyms and meronyms, as in many cases the holonym is more general than the meronym (Shwartz et al., 2016).⁷ The

⁶We considered at least 10 measures, allowing scores slightly lower than 0.8 when others were unavailable.

⁷In the hypernymy dataset of Shwartz et al. (2016),

surprising result could be explained by the nature of meronymy in this dataset: most holonyms in BLESS are rather specific words.

BLESS was built starting from 200 basic level concepts (e.g. *goldfish*) used as the x words, to which y words in different relations were associated (e.g. *eye*, for meronymy; *animal*, for hypernymy). x words represent hyponyms in the hyponym-hypernym pairs, and should therefore not be too general. Indeed, SLQS assigns high scores to hyponym-hypernym pairs. At the same time, in the meronymy relation in BLESS, x is the holonym and y is the meronym. For consistency with EVALution, we switched those pairs in BLESS, placing the meronym in the x slot and the holonym in the y slot. As a consequence, after the switching, holonyms in BLESS are usually rather specific words (e.g., there are no holonyms like *animal* and *vehicle*, as these words were originally in the y slot). In most cases, they are not more general than their meronyms (*(eye, goldfish)*), yielding low SLQS scores which are easy to separate from hypernyms. We note that this is a weakness of the BLESS dataset, rather than a strength of the measure. For instance, on EVALution, SLQS performs worse (ranked only as high as 13th), as this dataset has no such restriction on the basic level concepts, and may contain pairs like *(eye, animal)*.

Hypernym vs. Attribute Symmetric similarity measures computed on syntactic contexts succeed to discriminate between hypernyms and attributes. Since attributes are syntactically different from hypernyms (in attributes, y is an adjective), it is unsurprising that they occur in different syntactic contexts, yielding low similarity scores.

nearly 50% of the SLQS false positive pairs were meronym-holonym pairs, in many of which the holonym is more general than the meronym by definition, e.g. *(mauritus, africa)*.

dataset	hyper vs. relation	best supervised				best unsupervised			
		method	vectors	penalty	AP @100	measure	context type	feature weighting	AP @100
EVALution	meronym	concat	dep-based	L_2	0.998	APSyn	joint	freq	0.886
	attribute	concat	Glove-100	L_2	1.000	invCL	dep	ppmi	0.877
	antonym	concat	dep-based	L_2	1.000	invCL	joint	ppmi	0.773
	synonym	concat	dep-based	L_1	0.996	SLQS _{sub}	win2	plmi	0.813
BLESS	meronym	concat	Glove-50	L_1	1.000	SLQS _{sub}	win5	freq	0.939
	coord	concat	Glove-300	L_1	1.000	SLQS _{row_{sub}}	joint	plmi	0.938
	attribute	concat	Glove-100	L_1	1.000	SLQS _{sub}	dep	freq	0.938
	event	concat	Glove-100	L_1	1.000	SLQS _{sub}	dep	freq	0.847
	random-n	concat	word2vec	L_1	0.995	cosWeeds	win2d	ppmi	0.818
	random-j	concat	Glove-200	L_1	1.000	SLQS _{sub}	dep	freq	0.917
random-v	concat	word2vec	L_1	1.000	SLQS _{sub}	dep	freq	0.895	
Lenci/ Benotto	antonym	concat	dep-based	L_2	0.917	invCL	joint	ppmi	0.807
	synonym	concat	Glove-300	L_1	0.946	invCL	win5d	freq	0.914
Weeds	coord	concat	dep-based	L_2	0.873	invCL	win2d	freq	0.824
						SLQS _{row_{sub}}	joint	ppmi	

Table 4: Best performance on the validation set (10%) of each dataset for the supervised and unsupervised measures, in terms of Average Precision (AP) at $k = 100$, for hypernym vs. each single relation.

Hypernym vs. Antonym In all our experiments, antonyms were the hardest to distinguish from hypernyms, yielding the lowest performance. We found that SLQS performed reasonably well in this setting. However, the measure variations, context types and feature weightings were not consistent across datasets. SLQS relies on the assumption that y is a more general word than x , which is not true for antonyms, making it the most suitable measure for this setting.

Hypernym vs. Synonym SLQS performs well also in discriminating between hypernyms and synonyms, in which y is also not more general than x . We observed that in the `joint` context type, the difference in SLQS scores between synonyms and hypernyms was the largest. This may stem from the restrictiveness of this context type. For instance, among the most salient contexts we would expect to find informative contexts like *drinks milk* for *cat* and less informative ones like *drinks water* for *animal*, whereas the non-restrictive single dependency context *drinks* would probably be present for both.

Another measure that works well is `invCL`: interestingly, other inclusion-based measures assign high scores to (x, y) when y includes many of x 's contexts, which might be true also for synonyms (e.g. *elevator* and *lift* share many contexts). `invCL`, on the other hand, reduces with the ratio of y 's contexts included in x , yielding lower scores for synonyms.

Hypernym vs. Coordination We found no consistency among BLESS and Weeds. On Weeds,

inclusion-based measures (`ClarkeDE`, `invCL` and `Weeds`) showed the best results. The best performing measures on BLESS, however, were variants of SLQS, that showed to perform well in cases where the negative relation is symmetric (antonym, synonym and coordination). The difference could be explained by the nature of the datasets: the BLESS test set contains 1,185 hypernymy pairs, with only 129 distinct ys , many of which are general words like *animal* and *object*. The Weeds test set, on the other hand, was intentionally constructed to contain an overall unique y in each pair, and therefore contains much more specific ys (e.g. (*quirk*, *strangeness*)). For this reason, generality-based measures perform well on BLESS, and struggle with Weeds, which is handled better using inclusion-based measures.

5.3 Comparison to State-of-the-art Supervised Methods

For comparison with the state-of-the-art, we evaluated several supervised hypernymy detection methods, based on the word embeddings of x and y : concatenation $\vec{v}_x \oplus \vec{v}_y$ (Baroni et al., 2012), difference $\vec{v}_y - \vec{v}_x$ (Weeds et al., 2014), and ASYM (Roller et al., 2014). We downloaded several pre-trained embeddings (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014), and trained a logistic regression classifier to predict hypernymy. We used the 90% portion (originally the test set) as the train set, and the other 10% (originally the validation set) as a test set, reporting the best results among different vectors,

	method	AP@100 original	AP@100 switched	Δ
supervised	concat, word2vec, L_1	0.995	0.575	-0.42
unsupervised	cosWeeds, win2d, ppmi	0.818	0.882	+0.064

Table 5: Average Precision (AP) at $k = 100$ of the best supervised and unsupervised methods for hypernym vs. random-n, on the original BLESS validation set and the validation set with the artificially added switched hypernym pairs.

method and regularization factor.⁸

Table 4 displays the performance of the best classifier on each dataset, in a hypernym vs. a single relation setting. We also re-evaluated the unsupervised measures, this time reporting the results on the validation set (10%) for comparison.

The overall performance of the embedding-based classifiers is almost perfect, and in particular the best performance is achieved using the concatenation method (Baroni et al., 2012) with either GloVe (Pennington et al., 2014) or the dependency-based embeddings (Levy and Goldberg, 2014). As expected, the unsupervised measures perform worse than the embedding-based classifiers, though generally not bad on their own.

These results may suggest that unsupervised methods should be preferred only when no training data is available, leaving all the other cases to supervised methods. This is, however, not completely true. As others previously noticed, supervised methods do not actually learn the relation between x and y , but rather separate properties of either x or y . Levy et al. (2015b) named this the “lexical memorization” effect, i.e. memorizing that certain y s tend to appear in many positive pairs (*prototypical hypernyms*).

On that account, the Weeds dataset has been designed to avoid such memorization, with every word occurring once in each slot of the relation. While the performance of the supervised methods on this dataset is substantially lower than their performance on other datasets, it is yet well above the random baseline which we might expect from a method that can only memorize words it has seen during training.⁹ This is an indication that supervised methods can abstract away from the words.

Indeed, when we repeated the experiment with a lexical split of each dataset, i.e., such that the train and test set consist of distinct vocabularies, we found that the supervised methods’ performance did not decrease dramatically, in contrast to the

findings of Levy et al. (2015b). The large performance gaps reported by Levy et al. (2015b) might be attributed to the size of their training sets. Their lexical split discarded around half of the pairs in the dataset and split the rest of the pairs equally to train and test, resulting in a relatively small train set. We performed the split such that only around 30% of the pairs in each dataset were discarded, and split the train and test sets with a ratio of roughly 90/10%, obtaining large enough train sets.

Our experiment suggests that rather than memorizing the verbatim *prototypical hypernyms*, the supervised models might learn that certain regions in the vector space pertain to prototypical hypernyms. For example, *device* (from the BLESS train set) and *appliance* (from the BLESS test set) are two similar words, which are both prototypical hypernyms. Another interesting observation was recently made by Roller and Erk (2016): they showed that when dependency-based embeddings are used, supervised distributional methods trace x and y ’s separate occurrences in different slots of Hearst patterns (Hearst, 1992).

Whether supervised methods only memorize or also learn, it is more consensual that they lack the ability to capture the relation between x and y , and that they rather indicate how likely y (x) is to be a hypernym (hyponym) (Levy et al., 2015b; Santus et al., 2016a; Shwartz et al., 2016; Roller and Erk, 2016). While this information is valuable, it cannot be solely relied upon for classification.

To better understand the extent of this limitation, we conducted an experiment in a similar manner to the switched hypernym pairs in Santus et al. (2016a). We used BLESS, which is the only dataset with random pairs. For each hypernym pair (x_1, y_1) , we sampled a word y_2 that participates in another hypernym pair (x_2, y_2) , such that (x_1, y_2) is not in the dataset, and added (x_1, y_2) as a random pair. We added 139 new pairs to the validation set, such as (*rifle, animal*) and (*salmon, weapon*). We then used the best supervised and unsupervised methods for hypernym vs. random-n on BLESS to re-classify the revised validation set. Table 5 displays the experiment results.

⁸In our preliminary experiments we also trained other classifiers used in the distributional hypernymy detection literature (SVM and SVM+RBF kernel), that performed similarly. We report the results for logistic regression, since we use the prediction probabilities to measure average precision.

⁹The dataset is balanced between its two classes.

The switched hypernym experiment paints a much less optimistic picture of the embeddings’ actual performance, with a drop of 42 points in average precision. 121 out of the 139 switched hypernym pairs were falsely classified as hypernyms. Examining the y words of these pairs reveals general words that appear in many hypernym pairs (e.g. *animal*, *object*, *vehicle*). The unsupervised measure was not similarly affected by the switched pairs, and the performance even slightly increased. This result is not surprising, since most unsupervised measures aim to capture aspects of the relation between x and y , while not relying on information about one of the words in the pair.¹⁰

6 Discussion

The results in Section 5 suggest that a supervised method using the unsupervised measures as features could possibly be the best of both worlds. We would expect it to be more robust than embedding-based methods on the one hand, while being more informative than any single unsupervised measure on the other hand.

Such a method was developed by Santus et al. (2016a), however using mostly features that describe a single word, e.g. frequency and entropy. It was shown to be competitive with the state-of-the-art supervised methods. With that said, it was also shown to be sensitive to the distribution of training examples in a specific dataset, like the embedding-based methods.

We conducted a similar experiment, with a much larger number of unsupervised features, namely the various measure scores, and encountered the same issue. While the performance was good, it dropped dramatically when the model was tested on a different test set.

We conjecture that the problem stems from the currently available datasets, which are all somewhat artificial and biased. Supervised methods which are strongly based on the relation between the words, e.g. those that rely on path-based information (Shwartz et al., 2016), manage to overcome the bias. Distributional methods, on the other hand, are based on a weaker notion of the relation between words, hence are more prone to overfit the distribution of training instances in a specific dataset. In the future, we hope that new

¹⁰Turney and Mohammad (2015) have also shown that unsupervised methods are more robust than supervised ones in a transfer-learning experiment, when the “training data” was used to tune their parameters.

datasets will be available for the task, which would be drawn from corpora and will reflect more realistic distributions of words and semantic relations.

7 Conclusion

We performed an extensive evaluation of unsupervised methods for discriminating hypernyms from other semantic relations. We found that there is no single combination of measure and parameters which is always preferred; however, we suggested a principled linguistic-based analysis of the most suitable measure for each task that yields consistent performance across different datasets.

We investigated several new variants of existing methods, and found that some variants of SLQS turned out to be superior on certain tasks. In addition, we have tested for the first time the `joint` context type (Chersoni et al., 2016), which was found to be very discriminative, and might hopefully benefit other semantic tasks.

For comparison, we evaluated the state-of-the-art supervised methods on the datasets, and they have shown to outperform the unsupervised ones, while also being efficient and easier to use. However, a deeper analysis of their performance demonstrated that, as previously suggested, these methods do not capture the relation between x and y , but rather indicate the “prior probability” of either word to be a hyponym or a hypernym. As a consequence, supervised methods are sensitive to the distribution of examples in a particular dataset, making them less reliable for real-world applications. Being motivated by linguistic hypotheses, and independent from training data, unsupervised measures were shown to be more robust. In this sense, unsupervised methods can still play a relevant role, especially if combined with supervised methods, in the decision whether the relation holds or not.

Acknowledgments

The authors would like to thank Ido Dagan, Alessandro Lenci, and Yuji Matsumoto for their help and advice. Vered Shwartz is partially supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1). Enrico Santus is partially supported by HK PhD Fellowship Scheme under PF12-13656.

References

- Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.
- Marco Baroni and Alessandro Lenci. 2011. Proceedings of the gems 2011 workshop on geometrical models of natural language semantics. In *How we BLESSed distributional semantic evaluation*, pages 1–10. Association for Computational Linguistics.
- Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.
- Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.
- Giulia Benotto. 2015. Distributional models for semantic relations: A study on hyponymy and antonymy. *PhD Thesis, University of Pisa*.
- John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526.
- Emmanuele Chersoni, Enrico Santus, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. 2016. Representing verbs with rich contexts: an evaluation on verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972. Association for Computational Linguistics.
- Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.
- Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119. Association for Computational Linguistics.
- Ido Dagan, Dan Roth, and Mark Sammons. 2013. *Recognizing textual entailment*. Morgan & Claypool Publishers.
- Stefan Evert. 2005. The statistics of word cooccurrences: word pairs and collocations. *Dissertation*.
- Stefan Evert. 2008. Corpora and collocations. *Corpus linguistics. An international handbook*, 2:223–233.
- Adriano Ferraresi. 2007. Building a very large corpus of english obtained by web crawling: ukwac. *Masters thesis, University of Bologna, Italy*.
- Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.
- Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*.
- Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015. Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 119–124. Association for Computational Linguistics.
- Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389.
- Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 75–79. Association for Computational Linguistics.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308. Association for Computational Linguistics.
- Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.
- Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976. Association for Computational Linguistics.
- Dekang Lin. 1998. An information-theoretic definition of similarity. *ICML*, 98:296–304.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositional-ity. In *Advances in neural information processing systems*, pages 3111–3119.
- Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of*

- the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 11, pages 1872–1877.
- Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.
- Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 511–519. Association for Computational Linguistics.
- Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2163–2172. Association for Computational Linguistics.
- Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036. Dublin City University and Association for Computational Linguistics.
- Gerard Salton and Michael J. McGill. 1986. *Introduction to modern information retrieval*. McGraw-Hill, Inc.
- Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 38–42. Association for Computational Linguistics.
- Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Proceedings of the 4th workshop on linked data in linguistics: Resources and applications. In *EVALution 1.0: an Evolving Semantic Dataset for Training and Evaluation of Distributional Semantic Models*, pages 64–69. Association for Computational Linguistics.
- Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016a. Nine features in a random forest to learn taxonomical semantic relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4557–4564. European Language Resources Association (ELRA).
- Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016b. Unsupervised measure of word similarity: How to outperform co-occurrence and vector cosine in vsms. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 4260–4261.
- Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398. Association for Computational Linguistics.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics.
- Peter D. Turney and Saif M. Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(03):437–476.
- Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 81–88.
- Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259. Dublin City University and Association for Computational Linguistics.

Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network

Kim Anh Nguyen and Sabine Schulte im Walde and Ngoc Thang Vu

Institut für Maschinelle Sprachverarbeitung

Universität Stuttgart

Pfaffenwaldring 5B, 70569 Stuttgart, Germany

{nguyenkh, schulte, thangvu}@ims.uni-stuttgart.de

Abstract

Distinguishing between antonyms and synonyms is a key task to achieve high performance in NLP systems. While they are notoriously difficult to distinguish by distributional co-occurrence models, pattern-based methods have proven effective to differentiate between the relations. In this paper, we present a novel neural network model *AntSynNET* that exploits lexico-syntactic patterns from syntactic parse trees. In addition to the lexical and syntactic information, we successfully integrate the distance between the related words along the syntactic path as a new pattern feature. The results from classification experiments show that *AntSynNET* improves the performance over prior pattern-based methods.

1 Introduction

Antonymy and synonymy represent lexical semantic relations that are central to the organization of the mental lexicon (Miller and Fellbaum, 1991). While antonymy is defined as the oppositeness between words, synonymy refers to words that are similar in meaning (Deese, 1965; Lyons, 1977). From a computational point of view, distinguishing between antonymy and synonymy is important for NLP applications such as Machine Translation and Textual Entailment, which go beyond a general notion of semantic relatedness and require to identify specific semantic relations. However, due to interchangeable substitution, antonyms and synonyms often occur in similar contexts, which makes it challenging to automatically distinguish between them.

Two families of approaches to differentiate between antonyms and synonyms are predominant

in NLP. Both make use of distributional vector representations, relying on the *distributional hypothesis* (Harris, 1954; Firth, 1957), that words with similar distributions have related meanings: co-occurrence models and pattern-based models. These distributional semantic models (DSMs) offer a means to represent meaning vectors of words or word pairs, and to determine their semantic relatedness (Turney and Pantel, 2010).

In *co-occurrence models*, each word is represented by a weighted feature vector, where features typically correspond to words that co-occur in particular contexts. When using word embeddings, these models rely on neural methods to represent words as low-dimensional vectors. To create the word embeddings, the models either make use of neural-based techniques, such as the skip-gram model (Mikolov et al., 2013), or use matrix factorization (Pennington et al., 2014) that builds word embeddings by factorizing word-context co-occurrence matrices. In comparison to standard co-occurrence vector representations, word embeddings address the problematic sparsity of word vectors and have achieved impressive results in many NLP tasks such as word similarity (e.g., Pennington et al. (2014)), relation classification (e.g., Vu et al. (2016)), and antonym-synonym distinction (e.g., Nguyen et al. (2016)).

In *pattern-based models*, vector representations make use of lexico-syntactic surface patterns to distinguish between the relations of word pairs. For example, Justeson and Katz (1991) suggested that adjectival opposites co-occur with each other in specific linear sequences, such as between *X* and *Y*. Hearst (1992) determined surface patterns, e.g., *X such as Y*, to identify nominal hypernyms. Lin et al. (2003) proposed two textual patterns indicating semantic incompatibility, from *X to Y* and either *X or Y*, to distinguish opposites from semantically similar

words. Roth and Schulte im Walde (2014) proposed a method that combined patterns with discourse markers for classifying paradigmatic relations including antonymy, synonymy, and hypernymy. Recently, Schwartz et al. (2015) used two prominent patterns from Lin et al. (2003) to learn word embeddings that distinguished antonyms from similar words in determining degrees of similarity and word analogy.

In this paper, we present a novel pattern-based neural method *AntSynNET* to distinguish antonyms from synonyms. We hypothesize that antonymous word pairs co-occur with each other in lexico-syntactic patterns within a sentence more often than would be expected by synonymous pairs. This hypothesis is inspired by corpus-based studies on antonymy and synonymy. Among others, Charles and Miller (1989) suggested that adjectival opposites co-occur in patterns; Fellbaum (1995) stated that nominal and verbal opposites co-occur in the same sentence significantly more often than chance; Lin et al. (2003) argued that if two words appear in clear antonym patterns, they are unlikely to represent synonymous pair.

We start out by inducing patterns between X and Y from a large-scale web corpus, where X and Y represent two words of an antonym or synonym word pair, and the pattern is derived from the simple paths between X and Y in a syntactic parse tree. Each node in the simple path combines lexical and syntactic information; in addition, we suggest a novel feature for the patterns, i.e., the distance between the two words along the syntactic path. All pattern features are fed into a recurrent neural network with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997), which encode the patterns as vector representations. Afterwards, the vector representations of the patterns are used in a classifier to distinguish between antonyms and synonyms. The results from experiments show that *AntSynNET* improves the performance over prior pattern-based methods. Furthermore, the implementation of our models is made publicly available¹.

The remainder of this paper is organized as follows: In Section 2, we present previous work distinguishing antonyms and synonyms. Section 3 describes our proposed *AntSynNET* model. We present the induction of the patterns (Section 3.1), describe the recurrent neural network with long

short-term memory units which is used to encode patterns within a vector representation (Section 3.2), and describe two models to classify antonyms and synonyms: the pure pattern-based model (Section 3.3.1) and the combined model (Section 3.3.2). After introducing two baselines in Section 4, we describe our dataset, experimental settings, results of our methods, the effects of the newly proposed distance feature, and the effects of the various types of word embeddings. Section 6 concludes the paper.

2 Related Work

Pattern-based methods: Regarding the task of antonym-synonym distinction, there exist a variety of approaches which rely on patterns. Lin et al. (2003) used bilingual dependency triples and patterns to extract distributionally similar words. They relied on clear antonym patterns such as *from X to Y* and *either X or Y* in a post-processing step to distinguish antonyms from synonyms. The main idea is that if two words X and Y appear in one of these patterns, they are unlikely to represent synonymous pair. Schulte im Walde and Köper (2013) proposed a method to distinguish between the paradigmatic relations antonymy, synonymy and hypernymy in German, based on automatically acquired word patterns. Roth and Schulte im Walde (2014) combined general lexico-syntactic patterns with discourse markers as indicators for the same relations, both for German and for English. They assumed that if two phrases frequently co-occur with a specific discourse marker, then the discourse relation expressed by the corresponding marker should also indicate the relation between the words in the affected phrases. By using the raw corpus and a fixed list of discourse markers, the model can easily be extended to other languages. More recently, Schwartz et al. (2015) presented a symmetric pattern-based model for word vector representation in which antonyms are assigned to dissimilar vector representations. Differently to the previous pattern-based methods which used the standard distribution of patterns, Schwartz et al. used patterns to learn word embeddings.

Vector representation methods: Yih et al. (2012) introduced a new vector representation where antonyms lie on opposite sides of a sphere. They derived this representation with the incorporation of a thesaurus and latent semantic anal-

¹<https://github.com/nguyenkh/AntSynNET>

ysis, by assigning signs to the entries in the co-occurrence matrix on which latent semantic analysis operates, such that synonyms would tend to have positive cosine similarities, and antonyms would tend to have negative cosine similarities. Scheible et al. (2013) showed that the distributional difference between antonyms and synonyms can be identified via a simple word space model by using appropriate features. Instead of taking into account all words in a window of a certain size for feature extraction, the authors experimented with only words of a certain part-of-speech, and restricted distributions. Santus et al. (2014) proposed a different method to distinguish antonyms from synonyms by identifying the most salient dimensions of meaning in vector representations and reporting a new average-precision-based distributional measure and an entropy-based measure. Ono et al. (2015) trained supervised word embeddings for the task of identifying antonymy. They proposed two models to learn word embeddings: the first model relied on thesaurus information; the second model made use of distributional information and thesaurus information. More recently, Nguyen et al. (2016) proposed two methods to distinguish antonyms from synonyms: in the first method, the authors improved the quality of weighted feature vectors by strengthening those features that are most salient in the vectors, and by putting less emphasis on those that are of minor importance when distinguishing degrees of similarity between words. In the second method, the lexical contrast information was integrated into the skip-gram model (Mikolov et al., 2013) to learn word embeddings. This model successfully predicted degrees of similarity and identified antonyms and synonyms.

3 AntSynNET: LSTM-based Antonym-Synonym Distinction

In this section, we describe the AntSynNET model, using a pattern-based LSTM for distinguishing antonyms from synonyms. We first present the induction of patterns from a parsed corpus (Section 3.1). Section 3.2 then describes how we utilize the recurrent neural network with long short-term memory units to encode the patterns as vector representation. Finally, we present the AntSynNET model and two approaches to classify antonyms and synonyms (Section 3.3).

3.1 Induction of Patterns

Corpus-based studies on antonymy have suggested that opposites co-occur with each other within a sentence significantly more often than would be expected by chance. Our method thus makes use of patterns as the main indicators of word pair co-occurrence, to enforce a distinction between antonyms and synonyms. Figure 1 shows a syntactic parse tree of the sentence “*My old village has been provided with the new services*”. Following the characterizations of a tree in graph theory, any two nodes (vertices) of a tree are connected by a simple path (or one unique path). The simple path is the shortest path between any two nodes in a tree and does not contain repeated nodes. In the example, the lexico-syntactic tree pattern of the antonymous pair *old*–*new* is determined by finding the simple path (in red) from the lemma `old` to the lemma `new`. It focuses on the most relevant information and ignores irrelevant information which does not appear in the simple path (i.e., *has*, *been*). The example pattern between $X = \text{old}$ and $Y = \text{new}$ in Figure 1 is represented as follows: `X/JJ/amod/2 -- village/NN/nsubj/1 -- provide/VBN/ROOT/0 -- with/IN/prep/1 -- service/NNS/pobj/2 -- Y/JJ/amod/3.`

Node Representation: The path patterns make use of four features to represent each node in the syntax tree: lemma, part-of-speech (POS) tag, dependency label and distance label. The lemma feature captures the lexical information of words in the sentence, while the POS and dependency features capture the morpho-syntactic information of the sentence. The distance label measures the path distance between the target word nodes in the syntactic tree. Each step between a parent and a child node represents a distance of 1; and the ancestor nodes of the remaining nodes in the path are represented by a distance of 0. For example, the node `provided` is an ancestor node of the simple path from `old` to `new`. The distances from the node `provided` to the nodes `village` and `old` are 1 and 2, respectively.

The vector representation of each node concatenates the four-feature vectors as follows:

$$\vec{v}_{node} = [\vec{v}_{lemma} \oplus \vec{v}_{pos} \oplus \vec{v}_{dep} \oplus \vec{v}_{dist}]$$

where \vec{v}_{lemma} , \vec{v}_{pos} , \vec{v}_{dep} , \vec{v}_{dist} represent the embeddings of the lemma, POS tag, dependency label

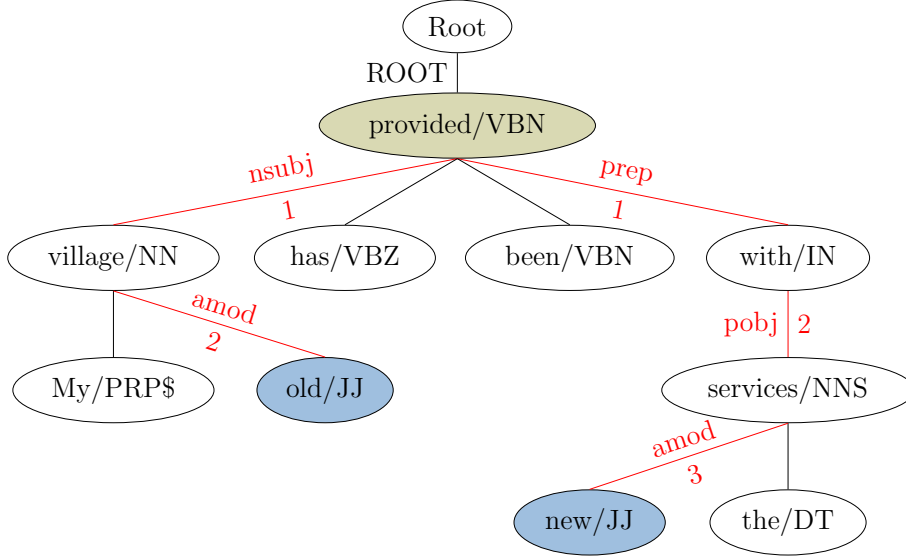


Figure 1: Illustration of the syntactic tree for the sentence “My old village has been provided with the new services”. Red lines indicate the path from the word `old` to the word `new`.

and distance label, respectively; and the \oplus denotes the concatenation operation.

Pattern Representation: For a pattern p which is constructed by the sequence of nodes n_1, n_2, \dots, n_k , the pattern representation of p is a sequence of vectors: $p = [\vec{n}_1, \vec{n}_2, \dots, \vec{n}_k]$. The pattern vector \vec{v}_p is then encoded by applying a recurrent neural network.

3.2 Recurrent Neural Network with Long Short-Term Memory Units

A recurrent neural network (RNN) is suitable for modeling sequential data by a vector representation. In our methods, we use a long short-term memory (LSTM) network, a variant of a recurrent neural network to encode patterns, for the following reasons. Given a sequence of words $p = [n_1, n_2, \dots, n_k]$ as input data, an RNN processes each word n_t at a time, and returns a vector of state h_k for the complete input sequence. For each time step t , the RNN updates an internal memory state h_t which depends on the current input n_t and the previous state h_{t-1} . Yet, if the sequential input is a long-term dependency, an RNN faces the problem of gradient vanishing or exploding, leading to difficulties in training the model.

LSTM units address these problems. The underlying idea of an LSTM is to use an adaptive gating mechanism to decide on the degree that LSTM units keep the previous state and memorize the extracted features of the current input. More specif-

ically, an LSTM comprises four components: an input gate i_t , a forget gate f_t , an output gate o_t , and a memory cell c_t . The state of an LSTM at each time step t is formalized as follows:

$$\begin{aligned} i_t &= \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \\ f_t &= \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \\ o_t &= \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \\ g_t &= \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \\ c_t &= i_t \otimes g_t + f_t \otimes c_{t-1} \end{aligned}$$

W refers to a matrix of weights that projects information between two layers; b is a layer-specific vector of bias terms; σ denotes the sigmoid function. The output of an LSTM at a time step t is computed as follows:

$$h_t = o_t \otimes \tanh(c_t)$$

where \otimes denotes element-wise multiplication. In our methods, we rely on the last state h_k to represent the vector \vec{v}_p of a pattern $p = [\vec{n}_1, \vec{n}_2, \dots, \vec{n}_k]$.

3.3 The Proposed AntSynNET Model

In this section, we present two models to distinguish antonyms from synonyms. The first model makes use of patterns to classify antonyms and synonyms, by using an LSTM to encode patterns as vector representations and then feeding those vectors to a logistic regression layer (Section 3.3.1). The second model creates combined vector representations of word pairs, which concatenate the vectors of the words and the patterns (Section 3.3.2).

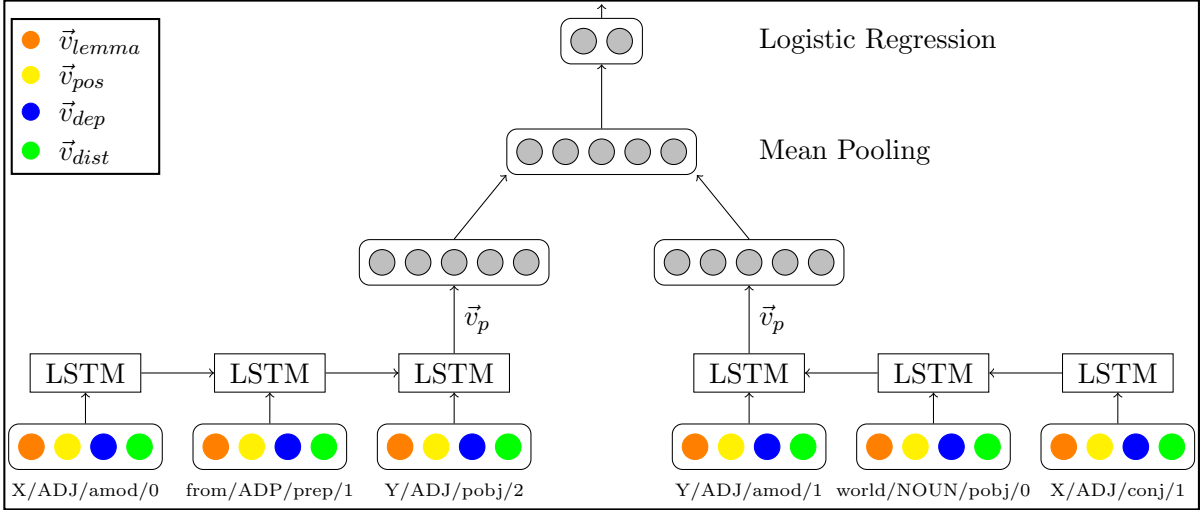


Figure 2: Illustration of the *AntSynNET* model. Each word pair is represented by several patterns, and each pattern represents a path in the graph of the syntactic tree. Patterns consist of several nodes where each node is represented by a vector with four features: lemma, POS, dependency label, and distance label. The mean pooling of the pattern vectors is the vector representation of each word pair, which is then fed to the logistic regression layer to classify antonyms and synonyms.

3.3.1 Pattern-based AntSynNET

In this model, we make use of a recurrent neural network with LSTM units to encode patterns containing a sequence of nodes. Figure 2 illustrates the *AntSynNET* model. Given a word pair (x, y) , we induce patterns for (x, y) from a corpus, where each pattern represents a path from x to y (cf. Section 3.1). We then feed each pattern p of the word pair (x, y) into an LSTM to obtain \vec{v}_p , the vector representation of the pattern p (cf. Section 3.2). For each word pair (x, y) , the vector representation of (x, y) is computed as follows:

$$\vec{v}_{xy} = \frac{\sum_{p \in P(x,y)} \vec{v}_p \cdot c_p}{\sum_{p \in P(x,y)} c_p} \quad (1)$$

\vec{v}_{xy} refers to the vector of the word pair (x, y) ; $P(x, y)$ is the set of patterns corresponding to the pair (x, y) ; c_p is the frequency of the pattern p . The vector \vec{v}_{xy} is then fed into a logistic regression layer whose target is the class label associated with the pair (x, y) . Finally, the pair (x, y) is predicted as positive (i.e., antonymous) word pair if the probability of the prediction for \vec{v}_{xy} is larger than 0.5.

3.3.2 Combined AntSynNET

Inspired by the supervised distributional concatenation method in Baroni et al. (2012) and the integrated path-based and distributional method for hypernymy detection in Shwartz et al. (2016), we

take into account the patterns and distribution of target pairs to create their combined vector representations. Given a word pair (x, y) , the combined vector representation of the pair (x, y) is determined by using both the co-occurrence distribution of the words and the syntactic path patterns:

$$\vec{v}_{comb(x,y)} = [\vec{v}_x \oplus \vec{v}_{xy} \oplus \vec{v}_y] \quad (2)$$

$\vec{v}_{comb(x,y)}$ refers to the combined vector of the word pair (x, y) ; \vec{v}_x and \vec{v}_y are the vectors of word x and word y , respectively; \vec{v}_{xy} is the vector of the pattern that corresponds to the pair (x, y) , cf. Section 3.3.1. Similar to the pattern-based model, the combined vector $\vec{v}_{comb(x,y)}$ is fed into the logistic regression layer to classify antonyms and synonyms.

4 Baseline Models

To compare *AntSynNET* with baseline models for pattern-based classification of antonyms and synonyms, we introduce two pattern-based baseline methods: the distributional method (Section 4.1), and the distributed method (Section 4.2).

4.1 Distributional Baseline

As a first baseline, we apply the approach by Roth and Schulte im Walde (2014), henceforth R&SiW. They used a vector space model to represent pairs of words by a combination of standard lexico-

syntactic patterns and discourse markers. In addition to the patterns, the discourse markers added information to express discourse relations, which in turn may indicate the specific semantic relation between the two words in a word pair. For example, contrast relations might indicate antonymy, whereas elaborations may indicate synonymy or hyponymy.

Michael Roth, the first author of R&SiW, kindly computed the relation classification results of the pattern–discourse model for our test sets. The weights between marker-based and pattern-based models were tuned on the validation sets; other hyperparameters were set exactly as described by the R&SiW method.

4.2 Distributed Baseline

The *SP* method proposed by Schwartz et al. (2015) uses symmetric patterns for generating word embeddings. In this work, the authors applied an unsupervised algorithm for the automatic extraction of symmetric patterns from plain text. The symmetric patterns were defined as a sequence of 3-5 tokens consisting of exactly two wildcards and 1-3 words. The patterns were filtered based on their frequencies, such that the resulting pattern set contained 11 patterns. For generating word embeddings, a matrix of co-occurrence counts between patterns and words in the vocabulary was computed, using positive point-wise mutual information. The sparsity problem of vector representations was addressed by smoothing. For antonym representation, the authors relied on two patterns suggested by Lin et al. (2003) to construct word embeddings containing an antonym parameter that can be turned on in order to represent antonyms as dissimilar, and that can be turned off to represent antonyms as similar.

To apply the *SP* method to our data, we make use of the pre-trained *SP* embeddings² with 500 dimensions³. We calculate the cosine similarity of word pairs and then use a Support Vector Machine with Radial Basis Function kernel to classify antonyms and synonyms.

²http://homes.cs.washington.edu/~roysch/papers/sp_embeddings/sp_embeddings.html

³The 500-dimensional embeddings outperformed the 300-dimensional embeddings for our data.

5 Experiments

5.1 Dataset

For training the models, neural networks require a large amount of training data. We use the existing large-scale antonym and synonym pairs previously used by Nguyen et al. (2016). Originally, the data pairs were collected from WordNet (Miller, 1995) and Wordnik⁴.

In order to induce patterns for the word pairs in the dataset, we identify the sentences in the corpus that contain the word pair. Thereafter, we extract all patterns for the word pair. We filter out all patterns which occur less than five times; and we only take into account word pairs that contain at least five patterns for training, validating and testing. For the proportion of positive and negative pairs, we keep a ratio of 1:1 positive (antonym) to negative (synonym) pairs in the dataset. In order to create the sets of training, testing and validation data, we perform random splitting with 70% train, 25% test, and 5% validation sets. The final dataset contains the number of word pairs according to word classes described in Table 1. Moreover, Table 2 shows the average number of patterns for each word pair in our dataset.

Word Class	Train	Test	Validation	Total
Adjective	5562	1986	398	7946
Verb	2534	908	182	3624
Noun	2836	1020	206	4062

Table 1: Our dataset.

Word Class	Train	Test	Validation
Adjective	135	131	141
Verb	364	332	396
Noun	110	132	105

Table 2: Average number of patterns per word pair across word classes.

5.2 Experimental Settings

We use the English Wikipedia dump⁵ from June 2016 as the corpus resource for our methods and baselines. For parsing the corpus, we rely on spaCy⁶. For the lemma embeddings, we rely on the word embeddings of the dLCE

⁴<http://www.wordnik.com>

⁵<https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2>

⁶<https://spacy.io>

Model	Adjective			Verb			Noun		
	P	R	F ₁	P	R	F ₁	P	R	F ₁
SP baseline	0.730	0.706	0.718	0.560	0.609	0.584	0.625	0.393	0.482
R&SiW baseline	0.717	0.717	0.717	0.789	0.787	0.788	0.833	0.831	0.832
Pattern-based AntSynNET	0.764	0.788	0.776*	0.741	0.833	0.784	0.804	0.851	0.827
Combined AntSynNET	0.763	0.807	0.784*	0.743	0.815	0.777	0.816	0.898	0.855**

Table 3: Performance of the AntSynNET models in comparison to the baseline models.

Feature	Model	Adjective			Verb			Noun		
		P	R	F ₁	P	R	F ₁	P	R	F ₁
Direction	Pattern-based	0.752	0.755	0.753	0.734	0.819	0.774	0.800	0.825	0.813
	Combined	0.754	0.784	0.769	0.739	0.793	0.765	0.829	0.810	0.819
Distance	Pattern-based	0.764	0.788	0.776	0.741	0.833	0.784	0.804	0.851	0.827
	Combined	0.763	0.807	0.784**	0.743	0.815	0.777	0.816	0.898	0.855**

Table 4: Comparing the novel distance feature with Schwarz et al.’s direction feature, across word classes.

model⁷ (Nguyen et al., 2016) which is the state-of-the-art vector representation for distinguishing antonyms from synonyms. We re-implemented this cutting-edge model on Wikipedia with 100 dimensions, and then make use of the dLCE word embeddings for initialization the lemma embeddings. The embeddings of POS tags, dependency labels, distance labels, and out-of-vocabulary lemmas are initialized randomly. The number of dimensions is set to 10 for the embeddings of POS tags, dependency labels and distance labels. We use the validation sets to tune the number of dimensions for these labels. For optimization, we rely on the cross-entropy loss function and Stochastic Gradient Descent with the Adadelta update rule (Zeiler, 2012). For training, we use the Theano framework (Theano Development Team, 2016). Regularization is applied by a dropout of 0.5 on each of component’s embeddings (dropout rate is tuned on the validation set). We train the models with 40 epochs and update all embeddings during training.

5.3 Overall Results

Table 3 shows the significant⁸ performance of our models in comparison to the baselines. Concerning adjectives, the two proposed models significantly outperform the two baselines: The performance of the baselines is around .72 for F_1 , and the corresponding results for the combined AntSynNET model achieve an improvement of $>.06$. Regarding nouns, the improvement of the new methods is just .02 F_1 in comparison to the

R&SiW baseline, but we achieve a much better performance in comparison to the SP baseline, an increase of .37 F_1 . Regarding verbs, we do not outperform the more advanced R&SiW baseline in terms of the F_1 score, but we obtain higher recall scores. In comparison to the SP baseline, our models still show a clear F_1 improvement.

Overall, our proposed models achieve comparatively high recall scores compared to the two baselines. This strengthens our hypothesis that there is a higher possibility for the co-occurrence of antonymous pairs in patterns over synonymous pairs within a sentence. Because, when the proposed models obtain high recall scores, the models are able to retrieve most relevant information (antonymous pairs) corresponding to the patterns. Regarding the low precision in the two proposed models, we sampled randomly 5 pairs in each population: true positive, true negative, false positive, false negative. We then compared the overlap of patterns for the true predictions (true positive pairs and true negative pairs) and the false predictions (false positive pairs and false negative pairs). We found out that there is no overlap between patterns of true predictions; and the number overlap between patterns of false predictions is 2, 2, and 4 patterns for noun, adjective, and verb classes, respectively. This shows that the low precision of our models stems from the patterns which represent both antonymous and synonymous pairs.

5.4 Effect of the Distance Feature

In our models, the novel distance feature is successfully integrated along the syntactic path to represent lexico-syntactic patterns. The intu-

⁷<https://github.com/nguyenkh/AntSynDistinction>

⁸t-test, * $p < 0.05$, ** $p < 0.1$

Model	Word Embeddings	Adjective			Verb			Noun		
		P	R	F ₁	P	R	F ₁	P	R	F ₁
Pattern-based Model	GloVe	0.763	0.770	0.767	0.705	0.852	0.772	0.789	0.849	0.818
	dLCE	0.764	0.788	0.776	0.741	0.833	0.784	0.804	0.851	0.827
Combined Model	Glove	0.750	0.798	0.773	0.717	0.826	0.768	0.807	0.827	0.817
	dLCE	0.763	0.807	0.784	0.743	0.815	0.777	0.816	0.898	0.855

Table 5: Comparing pre-trained GloVe and dLCE word embeddings.

ition behind the distance feature exploits properties of trees in graph theory, which show that there exist differences in the degree of relationship between the parent node and the child nodes ($distance = 1$) and in the degree of relationship between the ancestor node and the descendant nodes ($distance > 1$). Hence, we use the distance feature to effectively capture these relationships.

In order to evaluate the effect of our novel distance feature, we compare the distance feature to the direction feature proposed by Shwartz et al. (2016). In their approach, the authors combined lemma, POS, dependency, and direction features for the task of hypernym detection. The direction feature represented the direction of the dependency label between two nodes in a path from X to Y .

For evaluation, we make use of the same information regarding dataset and patterns as in Section 5.3, and then replace the distance feature by the direction feature. The results are shown in Table 4. The distance feature enhances the performance of our proposed models more effectively than the direction feature does, across all word classes.

5.5 Effect of Word Embeddings

Our methods rely on the word embeddings of the dLCE model, state-of-the-art word embeddings for antonym-synonym distinction. Yet, the word embeddings of the dLCE model, i.e., supervised word embeddings, represent information collected from lexical resources. In order to evaluate the effect of these word embeddings on the performance of our models, we replace them by the pre-trained GloVe word embeddings⁹ with 100 dimensions, and compare the effects of the GloVe word embeddings and the dLCE word embeddings on the performance of the two proposed models.

Table 5 illustrates the performance of our two models on all word classes. The table shows that the dLCE word embeddings are better than the

pre-trained GloVe word embeddings, by around .01 F_1 for the pattern-based AntSynNET model and the combined AntSynNET model regarding adjective and verb pairs. Regarding noun pairs, the improvements of the dLCE word embeddings over pre-trained GloVe word embeddings achieve around .01 and .04 F_1 for the pattern-based model and the combined model, respectively.

6 Conclusion

In this paper, we presented a novel pattern-based neural method *AntSynNET* to distinguish antonyms from synonyms. We hypothesized that antonymous word pairs co-occur with each other in lexico-syntactic patterns within a sentence more often than synonymous word pairs.

The patterns were derived from the simple paths between semantically related words in a syntactic parse tree. In addition to lexical and syntactic information, we suggested a novel path distance feature. The AntSynNET model consists of two approaches to classify antonyms and synonyms. In the first approach, we used a recurrent neural network with long short-term memory units to encode the patterns as vector representations; in the second approach, we made use of the distribution and encoded patterns of the target pairs to generate combined vector representations. The resulting vectors of patterns in both approaches were fed into the logistic regression layer for classification.

Our proposed models significantly outperformed two baselines relying on previous work, mainly in terms of recall. Moreover, we demonstrated that the distance feature outperformed a previously suggested direction feature, and that our embeddings outperformed the state-of-the-art GloVe embeddings. Last but not least, our two proposed models only rely on corpus data, such that the models are easily applicable to other languages and relations.

⁹<http://www-nlp.stanford.edu/projects/glove/>

Acknowledgements

We would like to thank Michael Roth for helping us to compute the results of the R&SiW model on our dataset.

The research was supported by the Ministry of Education and Training of the Socialist Republic of Vietnam (Scholarship 977/QD-BGDDT; Kim-Anh Nguyen), the DFG Collaborative Research Centre SFB 732 (Kim-Anh Nguyen, Ngoc Thang Vu), and the DFG Heisenberg Fellowship SCHU-2580/1 (Sabine Schulte im Walde).

References

- Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32, Avignon, France.
- Walter G. Charles and George A. Miller. 1989. Contexts of antonymous adjectives. *Applied Psychology*, 10:357–375.
- James Deese. 1965. *The Structure of Associations in Language and Thought*. The John Hopkins Press, Baltimore, MD.
- Christiane Fellbaum. 1995. Co-occurrence and antonymy. *International Journal of Lexicography*, 8:281–303.
- John R. Firth. 1957. *Papers in Linguistics 1934-51*. Longmans, London, UK.
- Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.
- Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, France.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- John S. Justeson and Slava M. Katz. 1991. Co-occurrences of antonymous adjectives and their contexts. *Computational Linguistics*, 17:1–19.
- Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1492–1493, Acapulco, Mexico.
- John Lyons. 1977. *Semantics*, volume 1. Cambridge University Press.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 746–751, Atlanta, Georgia.
- George A. Miller and Christiane Fellbaum. 1991. Semantic networks of English. *Cognition*, 41:197–229.
- George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.
- Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany.
- Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 984–989, Denver, Colorado.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.
- Michael Roth and Sabine Schulte im Walde. 2014. Combining word patterns and discourse markers for paradigmatic relation classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 524–530, Baltimore, MD.
- Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 38–42, Gothenburg, Sweden.
- Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 489–497, Nagoya, Japan.
- Sabine Schulte im Walde and Maximilian Köper. 2013. Pattern-based distinction of paradigmatic relations for german nouns, verbs, adjectives. In *Proceedings of the 25th International Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 189–198, Darmstadt, Germany.

- Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the 19th Conference on Computational Language Learning (CoNLL)*, pages 258–267, Beijing, China.
- Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2389–2398, Berlin, Germany.
- Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.
- Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.
- Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 534–539.
- Wen-tau Yih, Geoffrey Zweig, and John C. Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 1212–1222, Jeju Island, Korea.
- Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation

Alexander Panchenko[‡], Eugen Ruppert[‡], Stefano Faralli[†],
Simone Paolo Ponzetto[†] and Chris Biemann[‡]

[‡]Language Technology Group, Computer Science Dept., University of Hamburg, Germany

[†]Web and Data Science Group, Computer Science Dept., University of Mannheim, Germany

{panchenko, ruppert, biemann}@informatik.uni-hamburg.de

{faralli, simone}@informatik.uni-mannheim.de

Abstract

The current trend in NLP is the use of highly opaque models, e.g. neural networks and word embeddings. While these models yield state-of-the-art results on a range of tasks, their drawback is poor interpretability. On the example of word sense induction and disambiguation (WSID), we show that it is possible to develop an interpretable model that matches the state-of-the-art models in accuracy. Namely, we present an unsupervised, knowledge-free WSID approach, which is interpretable at three levels: word sense inventory, sense feature representations, and disambiguation procedure. Experiments show that our model performs on par with state-of-the-art word sense embeddings and other unsupervised systems while offering the possibility to justify its decisions in human-readable form.

1 Introduction

A word sense disambiguation (WSD) system takes as input a target word t and its context C . The system returns an identifier of a word sense s_i from the word sense inventory $\{s_1, \dots, s_n\}$ of t , where the senses are typically defined manually in advance. Despite significant progress in methodology during the two last decades (Ide and Véronis, 1998; Agirre and Edmonds, 2007; Moro and Navigli, 2015), WSD is still not widespread in applications (Navigli, 2009), which indicates the need for further progress. The difficulty of the problem largely stems from the lack of domain-specific training data. A fixed sense inventory, such as the one of WordNet (Miller, 1995), may contain irrelevant senses for the given application and at the same time lack relevant domain-specific senses.

Word sense induction from domain-specific corpora is supposed to solve this problem. However, most approaches to word sense induction and disambiguation, e.g. (Schütze, 1998; Li and Jurafsky, 2015; Bartunov et al., 2016), rely on clustering methods and dense vector representations that make a WSD model uninterpretable as compared to knowledge-based WSD methods.

Interpretability of a statistical model is important as it lets us understand the reasons behind its predictions (Vellido et al., 2011; Freitas, 2014; Li et al., 2016). Interpretability of WSD models (1) lets a user understand why in the given context one observed a given sense (e.g., for educational applications); (2) performs a comprehensive analysis of correct and erroneous predictions, giving rise to improved disambiguation models.

The contribution of this paper is an interpretable unsupervised knowledge-free WSD method. The novelty of our method is in (1) a technique to disambiguation that relies on induced inventories as a pivot for learning sense feature representations, (2) a technique for making induced sense representations interpretable by labeling them with hypernyms and images.

Our method tackles the interpretability issue of the prior methods; it is interpretable at the levels of (1) sense inventory, (2) sense feature representation, and (3) disambiguation procedure. In contrast to word sense induction by context clustering (Schütze (1998), inter alia), our method constructs an explicit word sense inventory. The method yields performance comparable to the state-of-the-art unsupervised systems, including two methods based on word sense embeddings. An open source implementation of the method featuring a live demo of several pre-trained models is available online.¹

¹<http://www.jobimtext.org/wsd>

2 Related Work

Multiple designs of WSD systems were proposed (Agirre and Edmonds, 2007; Navigli, 2009). They vary according to the level of supervision and the amount of external knowledge used. Most current systems either make use of lexical resources and/or rely on an explicitly annotated sense corpus.

Supervised approaches use a sense-labeled corpus to train a model, usually building one sub-model per target word (Ng, 1997; Lee and Ng, 2002; Klein et al., 2002; Wee, 2010). The IMS system by Zhong and Ng (2010) provides an implementation of the supervised approach to WSD that yields state-of-the-art results. While supervised approaches demonstrate top performance in competitions, they require large amounts of sense-labeled examples per target word.

Knowledge-based approaches rely on a lexical resource that provides a sense inventory and features for disambiguation and vary from the classical Lesk (1986) algorithm that uses word definitions to the Babelfy (Moro et al., 2014) system that uses harnesses a multilingual lexical-semantic network. Classical examples of such approaches include (Banerjee and Pedersen, 2002; Pedersen et al., 2005; Miller et al., 2012). More recently, several methods were proposed to learn sense embeddings on the basis of the sense inventory of a lexical resource (Chen et al., 2014; Rothe and Schütze, 2015; Camacho-Collados et al., 2015; Iacobacci et al., 2015; Nieto Piña and Johansson, 2016).

Unsupervised knowledge-free approaches use neither handcrafted lexical resources nor hand-annotated sense-labeled corpora. Instead, they induce word sense inventories automatically from corpora. Unsupervised WSD methods fall into two main categories: context clustering and word ego-network clustering.

Context clustering approaches, e.g. (Pedersen and Bruce, 1997; Schütze, 1998), represent an instance usually by a vector that characterizes its context, where the definition of context can vary greatly. These vectors of each instance are then clustered. Multi-prototype extensions of the skip-gram model (Mikolov et al., 2013) that use no pre-defined sense inventory learn one embedding word vector per one word sense and are commonly fitted with a disambiguation mechanism (Huang et al., 2012; Tian et al., 2014; Neelakantan et al.,

2014; Bartunov et al., 2016; Li and Jurafsky, 2015; Pelevina et al., 2016). Comparisons of the *AdaGram* (Bartunov et al., 2016) to (Neelakantan et al., 2014) on three SemEval word sense induction and disambiguation datasets show the advantage of the former. For this reason, we use *AdaGram* as a representative of the state-of-the-art word sense embeddings in our experiments. In addition, we compare to SenseGram, an alternative sense embedding based approach by Pelevina et al. (2016). What makes the comparison to the later method interesting is that this approach is similar to ours, but instead of sparse representations the authors rely on word embeddings, making their approach less interpretable.

Word ego-network clustering methods (Lin, 1998; Pantel and Lin, 2002; Widdows and Dorow, 2002; Biemann, 2006; Hope and Keller, 2013) cluster graphs of words semantically related to the ambiguous word. An ego network consists of a single node (ego) together with the nodes they are connected to (alters) and all the edges among those alters (Everett and Borgatti, 2005). In our case, such a network is a local neighborhood of one word. Nodes of the ego-network can be (1) words semantically similar to the target word, as in our approach, or (2) context words relevant to the target, as in the *UoS* system (Hope and Keller, 2013). Graph edges represent semantic relations between words derived using corpus-based methods (e.g. distributional semantics) or gathered from dictionaries. The sense induction process using word graphs is explored by (Widdows and Dorow, 2002; Biemann, 2006; Hope and Keller, 2013). Disambiguation of instances is performed by assigning the sense with the highest overlap between the instance’s context words and the words of the sense cluster. Véronis (2004) compiles a corpus with contexts of polysemous nouns using a search engine. A word graph is built by drawing edges between co-occurring words in the gathered corpus, where edges below a certain similarity threshold were discarded. His HyperLex algorithm detects hubs of this graph, which are interpreted as word senses. Disambiguation in this experiment is performed by computing the distance between context words and hubs in this graph.

Di Marco and Navigli (2013) presents a comprehensive study of several graph-based WSI methods including Chinese Whispers, HyperLex, curvature clustering (Dorow et al., 2005). Besides,

authors propose two novel algorithms: Balanced Maximum Spanning Tree Clustering and Squares (B-MST), Triangles and Diamonds (SquaT++). To construct graphs, authors use first-order and second-order relations extracted from a background corpus as well as keywords from snippets. This research goes beyond intrinsic evaluations of induced senses and measures impact of the WSI in the context of an information retrieval via clustering and diversifying Web search results. Depending on the dataset, HyperLex, B-MST or Chinese-Whispers provided the best results.

Our system combines several of above ideas and adds features ensuring interpretability. Most notably, we use a word sense inventory based on clustering word similarities (Pantel and Lin, 2002); for disambiguation we rely on syntactic context features, co-occurrences (Hope and Keller, 2013) and language models (Yuret, 2012).

Interpretable approaches. The need in methods that interpret results of opaque statistical models is widely recognised (Vellido et al., 2011; Vellido et al., 2012; Freitas, 2014; Li et al., 2016; Park et al., 2016). An interpretable WSD system is expected to provide (1) a human-readable sense inventory, (2) human-readable reasons why in a given context c a given sense s_i was detected. Lexical resources, such as WordNet, solve the first problem by providing manually-crafted definitions of senses, examples of usage, hypernyms, and synonyms. The BabelNet (Navigli and Ponzetto, 2010) integrates all these sense representations, adding to them links to external resources, such as Wikipedia, topical category labels, and images representing the sense. The unsupervised models listed above do not feature any of these representations making them much less interpretable as compared to the knowledge-based models. Ruppert et al. (2015) proposed a system for visualising sense inventories derived in an unsupervised way using graph-based distributional semantics. Panchenko (2016) proposed a method for making sense inventory of word sense embeddings interpretable by mapping it to BabelNet.

Our approach was inspired by the knowledge-based system Babelfy (Moro et al., 2014). While the inventory of Babelfy is interpretable as it relies on BabelNet, the system provides no underlying reasons behind sense predictions. Our objective was to reach interpretability level of knowledge-based models within an unsupervised framework.

3 Method: Unsupervised Interpretable Word Sense Disambiguation

Our unsupervised word sense disambiguation method consist of the five steps illustrated in Figure 1: extraction of context features (Section 3.1); computing word and feature similarities (Section 3.2); word sense induction (Section 3.3); labeling of clusters with hypernyms and images (Section 3.4), disambiguation of words in context based on the induced inventory (Section 3.5), and finally interpretation of the model (Section 3.6). Feature similarity and co-occurrence computation steps (drawn with a dashed lines) are optional, since they did not consistently improve performance.

3.1 Extraction of Context Features

The goal of this step is to extract word-feature counts from the input corpus. In particular, we extract three types of features:

Dependency Features. These feature represents a word by a syntactic dependency such as “nn(•,writing)” or “prep.at(sit,•)”, extracted from the Stanford Dependencies (De Marneffe et al., 2006) obtained with the the PCFG model of the Stanford parser (Klein and Manning, 2003). Weights are computed using the Local Mutual Information (LMI) (Evert, 2005). One word is represented with 1000 most significant features.

Co-occurrence Features. This type of features represents a word by another word. We extract the list of words that significantly co-occur in a sentence with the target word in the input corpus based on the log-likelihood as word-feature weight (Dunning, 1993).

Language Model Feature. This type of features are based on a trigram model with Kneser-Ney smoothing (Kneser and Ney, 1995). In particular, a word is represented by (1) right and left context words, e.g. “office_•_and”, (2) two preceding words “new_office_•”, and (3) two succeeding words, e.g. “•_and_chairs”. We use the conditional probabilities of the resulting trigrams as word-feature weights.

3.2 Computing Word and Feature Similarities

The goal of this step is to build a graph of word similarities, such as (table, chair, 0.78). We used the *JoBimText* framework (Biemann and Riedl,

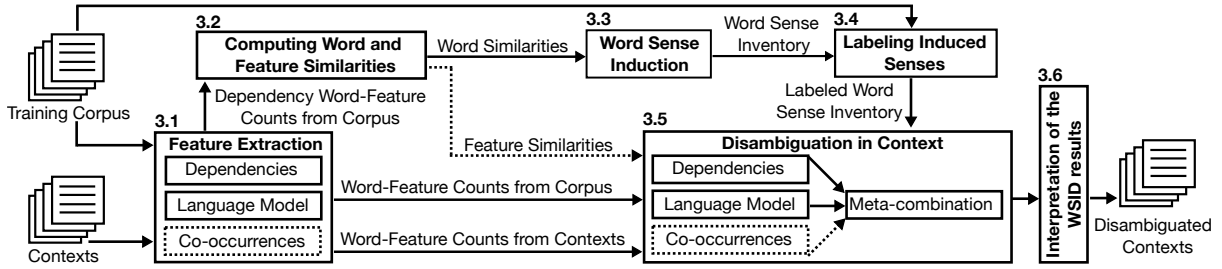


Figure 1: Outline of our unsupervised interpretable method for word sense induction and disambiguation.

2013) as it yields comparable performance on semantic similarity to state-of-the-art dense representations (Mikolov et al., 2013) compared on the WordNet as gold standard (Riedl, 2016), but is interpretable as words are represented by sparse interpretable features. Namely we use dependency-based features as, according to prior evaluations, this kind of features provides state-of-the-art semantic relatedness scores (Padó and Lapata, 2007; Van de Cruys, 2010; Panchenko and Morozova, 2012; Levy and Goldberg, 2014).

First, features of each word are ranked using the LMI metric (Evert, 2005). Second, the word representations are pruned keeping 1000 most salient features per word and 1000 most salient words per feature. The pruning reduces computational complexity and noise. Finally, word similarities are computed as a number of common features for two words. This is again followed by a pruning step in which only the 200 most similar terms are kept to every word. The resulting word similarities are browsable online.²

Note that while words can be characterized with distributions over features, features can vice versa be characterized by a distribution over words. We use this duality to compute feature similarities using the same mechanism and explore their use in disambiguation below.

3.3 Word Sense Induction

We induce a sense inventory by clustering of ego-network of similar words. In our case, an inventory represents senses by a word cluster, such as “chair, bed, bench, stool, sofa, desk, cabinet” for the “furniture” sense of the word “table”.

The sense induction processes one word t of the distributional thesaurus T per iteration. First, we retrieve nodes of the ego-network G of t being the N most similar words of t according to T (see

²Select the “JoBimViz” demo and then the “Stanford (English)” model: <http://www.jobimtext.org>.

Figure 2 (1)). Note that the target word t itself is not part of the ego-network. Second, we connect each node in G to its n most similar words according to T . Finally, the ego-network is clustered with Chinese Whispers (Biemann, 2006), a non-parametric algorithm that discovers the number of senses automatically. The n parameter regulates the granularity of the inventory: we experiment with $n \in \{200, 100, 50\}$ and $N = 200$.

The choice of Chinese Whispers among other algorithms, such as HyperLex (Véronis, 2004) or MCL (Van Dongen, 2008), was motivated by the absence of meta-parameters and its comparable performance on the WSI task to the state-of-the-art (Di Marco and Navigli, 2013).

3.4 Labeling Induced Senses with Hypernyms and Images

Each sense cluster is automatically labeled to improve its interpretability. First, we extract hypernyms from the input corpus using Hearst (1992) patterns. Second, we rank hypernyms relevant to the cluster by a product of two scores: the *hypernym relevance* score, calculated as $\sum_{w \in \text{cluster}} \text{sim}(t, w) \text{freq}(w, h)$, and the *hypernym coverage* score, calculated as $\sum_{w \in \text{cluster}} \min(\text{freq}(w, h), 1)$. Here the $\text{sim}(t, w)$ is the relatedness of the cluster word w to the target word t , and the $\text{freq}(w, h)$ is the frequency of the hypernymy relation (w, h) as extracted via patterns. Thus, a high-ranked hypernym h has high relevance, but also is confirmed by several cluster words. This stage results in a ranked list of labels that specify the word sense, for which we here show the first one, e.g. “table (furniture)” or “table (data)”.

Faralli and Navigli (2012) showed that web search engines can be used to bootstrap sense-related information. To further improve interpretability of induced senses, we assign an image to each word in the cluster (see Figure 2) by query-

ing the Bing image search API³ using the query composed of the target word and its hypernym, e.g. “jaguar car”. The first hit of this query is selected to represent the induced word sense.

Algorithm 1: Unsupervised WSD of the word t based on the induced word sense inventory I .

```

input : Word  $t$ , context features  $C$ , sense inventory  $I$ ,
        word-feature table  $F$ , use largest cluster
        back-off  $LCB$ , use feature expansion  $FE$ .
output: Sense of the target word  $t$  in inventory  $I$  and
        confidence score.
1  $S \leftarrow \text{getSenses}(I, t)$ 
2 if  $FE$  then
3   |  $C \leftarrow \text{featureExpansion}(C)$ 
4 end
5 foreach  $(sense, cluster) \in S$  do
6   |  $\alpha[sense] \leftarrow \{\}$ 
7   | foreach  $w \in cluster$  do
8     | foreach  $c \in C$  do
9       | |  $\alpha[sense] \leftarrow \alpha[sense] \cup F(w, c)$ 
10      | end
11     | end
12   | end
13 if  $\max_{sense \in S} \text{mean}(\alpha[sense]) = 0$  then
14   | if  $LCB$  then
15     | | return  $\arg \max_{(., cluster) \in S} |cluster|$ 
16     | else
17       | | return  $-1$  // reject to classify
18     | end
19   | else
20     | | return  $\arg \max_{(sense, .) \in S} \text{mean}(\alpha[sense])$ 
21   | end

```

3.5 Word Sense Disambiguation with Induced Word Sense Inventory

To disambiguate a target word t in context, we extract context features C and pass them to Algorithm 1. We use the induced sense inventory I and select the sense that has the largest weighted feature overlap with context features or fall back to the largest cluster back-off when context features C do not match the learned sense representations.

The algorithm starts by retrieving induced sense clusters of the target word (line 1). Next, the method starts to accumulate context feature weights of each $sense$ in $\alpha[sense]$. Each word w in a sense $cluster$ brings all its word-feature counts $F(w, c)$: see lines 5-12. Finally, a $sense$ that maximizes mean weight across all context features is chosen (lines 13-21). Optionally, we can resort to the largest cluster back-off (LCB) strategy in case if no context features match sense representations.

³<https://azure.microsoft.com/en-us/services/cognitive-services/search>

Note that the induced inventory I is used as a pivot to aggregate word-feature counts $F(w, c)$ of the words in the $cluster$ in order to build feature representations of each induced $sense$. We assume that the sets of similar words per sense are compatible with each other’s context. Thus, we can aggregate ambiguous feature representations of words in a sense cluster. In a way, occurrences of cluster members form the training set for the sense, i.e. contexts of {chair, bed, bench, stool, sofa, desk, cabinet}, add to the representation of “table (furniture)” in the model. Here, ambiguous cluster members like “chair” (which could also mean “chairman”) add some noise, but its influence is dwarfed by the aggregation over all cluster members. Besides, it is unlikely that the target (“table”) and the cluster member (“chair”) share the same homonymy, thus noisy context features hardly play a role when disambiguating the target in context. For instance, for scoring using language model features, we retrieve the context of the target word and substitute the target word one by one of the cluster words. To close the gap between the aggregated dependency per sense $\alpha[sense]$ and dependencies observed in the target’s context C , we use the similarity of features: we expand every feature $c \in C$ with 200 of most similar features and use them as additional features (lines 2-4).

We run disambiguation independently for each of the feature types listed above, e.g. dependencies or co-occurrences. Next, independent predictions are combined using the majority-voting rule.

3.6 Interpretability of the Method

Results of disambiguation can be interpreted by humans as illustrated by Figure 2. In particular, our approach is interpretable at three levels:

- 1. Word sense inventory.** To make induced word sense inventories interpretable we display senses of each word as an ego-network of its semantically related words. For instance, the network of the word “table” in our example is constructed from two tightly related groups of words that correspond to “furniture” and “data” senses. These labels of the clusters are obtained automatically (see Section 3.4).

While alternative methods, such as *AdaGram*, can generate sense clusters, our approach makes the senses better interpretable due to hypernyms and image labels that summarize senses.

skewed as 79% of contexts are assigned to the most frequent senses. Thus, in addition to the full TWSI dataset, we also use a balanced subset featuring five contexts per sense and 6,166 sentences to assess the quality of the disambiguation mechanism for smaller senses. This dataset contains no monosemous words to completely remove the bias of the most frequent sense. Note that de-biasing the evaluation set does not de-bias the word sense inventory, thus the task becomes harder for the balanced subset.

For the TWSI evaluation, we create an explicit mapping between the system-provided sense inventory and the TWSI word senses: senses are represented as the bag of words, which are compared using cosine similarity. Every induced sense gets assigned at most one TWSI sense. Once the mapping is completed, we calculate Precision, Recall, and F-measure. We use the following baselines to facilitate interpretation of the results: (1) MFS of the TWSI inventory always assigns the most frequent sense in the TWSI dataset; (2) LCB of the induced inventory always assigns the largest sense cluster; (3) Upper bound of the induced vocabulary always selects the correct sense for the context, but only if the mapping exists for this sense; (4) Random sense of the TWSI and the induced inventories.

4.1.2 Discussion of Results

The results of the TWSI evaluation are presented in Table 1. In accordance with prior art in word sense disambiguation, the most frequent sense (MFS) proved to be a strong baseline, reaching an F-score of 0.787, while the random sense over the TWSI inventory drops to 0.536. The upper bound on our induced inventory (F-score of 0.900) shows that the sense mapping technique used prior to evaluation does not drastically distort the evaluation scores. The LCB baseline of the induced inventory achieves an F-score of 0.691, demonstrating the efficiency of the LCB technique.

Let us first consider models based on single features. Dependency features yield the highest precision of 0.728, but have a moderate recall of 0.343 since they rarely match due to their sparsity. The LCB strategy for these rejected contexts helps to improve recall at cost of precision. Co-occurrence features yield significantly lower precision than the dependency-based features, but their recall is higher. Finally, the language model features yield very balanced results in terms of

both precision and recall. Yet, the precision of the model based on this feature type is significantly lower than that of dependencies.

Not all combinations improve results, e.g. combination of three types of features yields inferior results as compared to the language model alone. However, a combination of the language model with dependency features does provide an improvement over the single models as both these models bring strong signal of complementary nature about the semantics of the context. The dependency features represent syntactic information, while the LM features represent lexical information. This improvement is even more pronounced in the case of the balanced TWSI dataset. This combined model yields the best F-scores overall.

Table 2 presents the effect of the feature expansion based on the graph of similar features. For a low-recall model such the one based on syntactic dependencies, feature expansion makes a lot of sense: it almost doubles recall, while losing some precision. The gain in F-score using this technique is almost 20 points on the full TWSI dataset. However, the need for such expansion vanishes when two principally different types of features (precise syntactic dependencies and high-coverage trigram language model) are combined. Both precision and F-score of this combined model outperforms that of the dependency-based model with feature expansion by a large margin.

Figure 3 illustrates how granularity of the induced sense inventory influences WSD performance. For this experiment, we constructed three inventories, setting the number of most similar words in the ego-network n to 200, 100 and 50. These settings produced inventories with respectively 1.96, 2.98 and 5.21 average senses per target word. We observe that a higher sense granularity leads to lower F-scores. This can be explained because of (1) the fact that granularity of the TWSI is similar to granularity of the most coarse-grained inventory; (2) the higher the number of senses, the higher the chance to make a wrong sense assignment; (3) due to the reduced size of individual clusters, we get less signal per sense cluster and noise becomes more pronounced.

To summarize, the best precision is reached by a model based on un-expanded dependencies and the best F-score can be obtained by a combination of models based on un-expanded dependency features and language model features.

Model	#Senses	Full TWSI			Sense-Balanced TWSI		
		Prec.	Recall	F-score	Prec.	Recall	F-score
MFS of the TWSI inventory	2.31	0.787	0.787	0.787	0.373	0.373	0.373
Random Sense of the TWSI inventory	2.31	0.536	0.534	0.535	0.160	0.160	0.160
Upper bound of the induced inventory	1.96	1.000	0.819	0.900	1.000	0.598	0.748
Largest Cluster Back-Off (LCB) of the induced inventory	1.96	0.691	0.690	0.691	0.371	0.371	0.371
Random sense of the induced inventory	1.96	0.559	0.558	0.558	0.325	0.324	0.324
Dependencies	1.96	0.728	0.343	0.466	0.432	0.190	0.263
Dependencies + LCB	1.96	0.689	0.680	0.684	0.388	0.385	0.387
Co-occurrences (Cooc)	1.96	0.570	0.563	0.566	0.336	0.333	0.335
Language Model (LM)	1.96	0.685	0.677	0.681	0.416	0.412	0.414
Dependencies + LM + Cooc	1.96	0.644	0.636	0.640	0.388	0.386	0.387
Dependencies + LM	1.96	0.689	0.681	0.685	0.426	0.422	0.424

Table 1: WSD performance of different configurations of our method on the full and the sense-balanced TWSI datasets based on the coarse inventory with 1.96 senses/word ($N = 200, n = 200$).

Model	Precision	Recall	F-score	Precision	Recall	F-score
Dependencies	0.728	0.343	0.466	0.432	0.190	0.263
Dependencies Exp.	0.687	0.633	0.659	0.414	0.379	0.396
Dependencies + LM	0.689	0.681	0.685	0.426	0.422	0.424
Dependencies Exp. + LM	0.684	0.676	0.680	0.412	0.408	0.410

Table 2: Effect of the feature expansion: performance on the full (on the left) and the sense-balanced (on the right) TWSI datasets. The models were trained on the Wikipedia corpus using the coarse inventory (1.96 senses per word). The best results overall are underlined.

4.2 SemEval 2013 Task 13 Dataset

4.2.1 Dataset and Evaluation Metrics

The task of word sense induction for graded and non-graded senses provides 20 nouns, 20 verbs and 10 adjectives in WordNet-sense-tagged contexts. It contains 20-100 contexts per word, and 4,664 contexts in total with 6,73 sense per word in average. Participants were asked to cluster instances into groups corresponding to distinct word senses. Instances with multiple senses were labeled with a score between 0 and 1.

Performance is measured with three measures that require a mapping of inventories (Jaccard Index, Tau, WNDCG) and two cluster comparison measures (Fuzzy NMI, Fuzzy B-Cubed).

4.2.2 Discussion of Results

Table 3 presents results of evaluation of the best configuration of our approach trained on the ukWaC corpus. We compare our approach to four SemEval participants and two state-of-the-art systems based on word sense embeddings: *AdaGram* (Bartunov et al., 2016) based on Bayesian stick-breaking process⁵ and *SenseGram* (Pelevina et al., 2016) based on clustering of ego-network

generated using word embeddings⁶. The *AI-KU* system (Baskaya et al., 2013) directly clusters test contexts using the k -means algorithm based on lexical substitution features. The *Unimelb* system (Lau et al., 2013) uses one hierarchical topic model to induce and disambiguate senses of one word. The *UoS* system (Hope and Keller, 2013) induces senses by building an ego-network of a word using dependency relations, which is subsequently clustered using the MaxMax clustering algorithm. The *La Sapienza* system (Jurgens and Klapaftis, 2013), relies on WordNet for the sense inventory and disambiguation.

In contrast to the TWSI evaluation, the most fine-grained model yields the best scores, yet the inventory of the task is also more fine-grained than the one of the TWSI (7.08 vs. 2.31 avg. senses per word). Our method outperforms the knowledge-based system of *La Sapienza* according to two of three metrics metrics and the *SenseGram* system based on sense embeddings according to four of five metrics. Note that *SenseGram* outperforms all other systems according to the Fuzzy B-Cubed metric, which is maximized in the ‘‘All instances, One sense’’ settings. Thus this result may be due to

⁵<https://github.com/sbos/AdaGram.jl>

⁶<https://github.com/tudarmstadt-lt/sensegram>

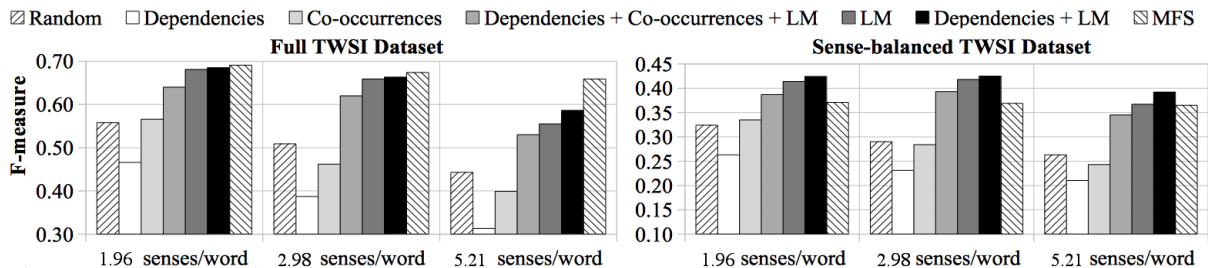


Figure 3: Impact of word sense inventory granularity on WSD performance: the TWSI dataset.

Model	Jacc. Ind.	Tau	WNDCG	Fuzzy NMI	Fuzzy B-Cubed
All Instances, One sense	0.192	0.609	0.288	0.000	0.623
1 sense per instance	0.000	0.953	0.000	0.072	0.000
Most Frequent Sense	0.552	0.560	0.412	–	–
AI-KU	0.197	0.620	0.387	0.065	0.390
AI-KU (remove5-add1000)	0.245	0.642	0.332	0.039	0.451
Unimelb (50k)	0.213	0.620	0.371	0.060	0.483
UoS (top-3)	0.232	0.625	0.374	0.045	0.448
La Sapienza (2)	0.149	0.510	0.383	–	–
AdaGram, $\alpha = 0.05$, 100 dim. vectors	0.274	0.644	0.318	0.058	0.470
SenseGram, 100 dim., CBOW, weight, sim., $p = 2$	0.197	0.615	0.291	0.011	0.615
Dependencies + LM (1.96 senses/word)	0.239	0.634	0.300	0.041	0.513
Dependencies + LM (2.98 senses/word)	0.242	0.634	0.300	0.041	0.504
Dependencies + LM (5.21 senses/word)	0.253	0.638	0.300	0.041	0.479

Table 3: WSD performance of the best configuration of our method identified on the TWSI dataset as compared to participants of the SemEval 2013 Task 13 and two systems based on word sense embeddings (AdaGram and SenseGram). All models were trained on the ukWaC corpus.

difference in granularities: the average polysemy of the *SenseGram* model is 1.56, while the polysemy of our models range from 1.96 to 5.21.

Besides, our system performs comparably to the top unsupervised systems participated in the competition: It is on par with the top SemEval submissions (*AI-KU* and *UoS*) and the another system based on embeddings (*AdaGram*), in terms of four out of five metrics (Jaccard Index, Tau, Fuzzy B-Cubed, Fuzzy NMI).

Therefore, we conclude that our system yields comparable results to the state-of-the-art unsupervised systems. Note, however, that none of the rivaling systems has a comparable level of interpretability to our approach. This is where our method is unique in the class of unsupervised methods: feature representations and disambiguation procedure of the neural-based *AdaGram* and *SenseGram* systems cannot be straightforwardly interpreted. Besides, inventories of the existing systems are represented as ranked lists of words lacking features that improve readability, such as hypernyms and images.

5 Conclusion

In this paper, we have presented a novel method for word sense induction and disambiguation that relies on a meta-combination of dependency features with a language model. The majority of existing unsupervised approaches focus on optimizing the accuracy of the method, sacrificing its interpretability due to the use of opaque models, such as neural networks. In contrast, our approach places a focus on interpretability with the help of sparse readable features. While being interpretable at three levels (sense inventory, sense representations and disambiguation), our method is competitive to the state-of-the-art, including two recent approaches based on sense embeddings, in a word sense induction task. Therefore, it is possible to match the performance of accurate, but opaque methods when interpretability matters.

Acknowledgments

We acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) foundation under the JOIN-T project.

References

- Eneko Agirre and Philip G. Edmonds. 2007. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.
- Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145, Mexico City, Mexico. Springer.
- Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.
- Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS'2016)*, Cadiz, Spain.
- Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. AI-KU: Using Substitute Vectors and Co-Occurrence Modeling for Word Sense Induction and Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 300–306, Atlanta, GA, USA. Association for Computational Linguistics.
- Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.
- Chris Biemann. 2006. Chinese Whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing*, pages 73–80, New York City, NY, USA. Association for Computational Linguistics.
- Chris Biemann. 2012. Turk Bootstrap Word Sense Inventory 2.0: A Large-Scale Resource for Lexical Substitution. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 4038–4042, Istanbul, Turkey. European Language Resources Association.
- José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, CO, USA. Association for Computational Linguistics.
- Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar. Association for Computational Linguistics.
- Marie-Catherine De Marneffe, Bill MacCartney, Christopher D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC'2006)*, pages 449–454, Genova, Italy. European Language Resources Association.
- Antonio Di Marco and Roberto Navigli. 2013. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, 39(3):709–754.
- Beate Dorow, Dominic Widdows, Katarina Ling, Jean-Pierre Eckmann, Danilo Sergi, and Elisha Moses. 2005. Using curvature and markov clustering in graphs for lexical acquisition and word sense discrimination. In *Proceedings of the Meaning-2005 Workshop*, Trento, Italy.
- Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19:61–74.
- Martin Everett and Stephen P. Borgatti. 2005. Ego network betweenness. *Social networks*, 27(1):31–38.
- Stefan Evert. 2005. *The Statistics of Word Cooccurrences Word Pairs and Collocations*. Ph.D. thesis, University of Stuttgart.
- Stefano Faralli and Roberto Navigli. 2012. A new minimally-supervised framework for domain word sense disambiguation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1411–1422, Jeju Island, Korea, July. Association for Computational Linguistics.
- Alex A. Freitas. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10.
- Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.
- David Hope and Bill Keller. 2013. MaxMax: A Graph-based Soft Clustering Algorithm Applied to Word Sense Induction. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 368–381, Samos, Greece. Springer.

- Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'2012)*, pages 873–882, Jeju Island, Korea. Association for Computational Linguistics.
- Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'2015)*, pages 95–105, Beijing, China. Association for Computational Linguistics.
- Nancy Ide and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational linguistics*, 24(1):2–40.
- David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 Task 13: Word Sense Induction for Graded and Non-graded Senses. In *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval'2013)*, pages 290–299, Montreal, Canada. Association for Computational Linguistics.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.
- Dan Klein, Kristina Toutanova, H. Tolga Ilhan, Sepandar D. Kamvar, and Christopher D. Manning. 2002. Combining Heterogeneous Classifiers for Word-Sense Disambiguation. In *Proceedings of the ACL'2002 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, volume 8, pages 74–80, Philadelphia, PA, USA. Association for Computational Linguistics.
- Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, volume 1, pages 181–184, Detroit, MI, USA. IEEE.
- Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic Modelling-based Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 307–311, Atlanta, GA, USA. Association for Computational Linguistics.
- Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'2002)*, volume 10, pages 41–48, Philadelphia, PA, USA. Association for Computational Linguistics.
- Michael Lesk. 1986. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, Toronto, ON, Canada. ACM.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, MD, USA. Association for Computational Linguistics.
- Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Conference on Empirical Methods in Natural Language Processing (EMNLP'2015)*, pages 1722–1732, Lisboa, Portugal. Association for Computational Linguistics.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, CA, USA. Association for Computational Linguistics.
- Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning (ICML'1998)*, volume 98, pages 296–304, Madison, WI, USA. Morgan Kaufmann Publishers Inc.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at International Conference on Learning Representations (ICLR)*, pages 1310–1318, Scottsdale, AZ, USA.
- Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1781–1796, Mumbai, India. Association for Computational Linguistics.
- George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.
- Andrea Moro and Roberto Navigli. 2015. SemEval-2015 Task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, CO, USA. Association for Computational Linguistics.

- Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.
- Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics*, pages 216–225, Uppsala, Sweden.
- Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.
- Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar. Association for Computational Linguistics.
- Hwee Tou Ng. 1997. Exemplar-Based Word Sense Disambiguation: Some Recent Improvements. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213, Providence, RI, USA. Association for Computational Linguistics.
- Luis Nieto Piña and Richard Johansson. 2016. Embedding senses for efficient graph-based word sense disambiguation. In *Proceedings of TextGraphs-10: the Workshop on Graph-based Methods for Natural Language Processing*, pages 1–5, San Diego, CA, USA. Association for Computational Linguistics.
- Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- Alexander Panchenko and Olga Morozova. 2012. A study of hybrid similarity measures for semantic relation extraction. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 10–18, Avignon, France. Association for Computational Linguistics.
- Alexander Panchenko. 2016. Best of both worlds: Making word sense embeddings interpretable. In *Proceedings of the 10th Language Resources and Evaluation Conference (LREC’2016)*, pages 2649–2655, Portoro, Slovenia. European Language Resources Association (ELRA).
- Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, Edmonton, AB, Canada.
- Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2016. Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757*.
- Ted Pedersen and Rebecca Bruce. 1997. Distinguishing word senses in untagged text. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP’1997)*, pages 197–207, Providence, RI, USA. Association for Computational Linguistics.
- Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. 2005. Maximizing semantic relatedness to perform word sense disambiguation. *University of Minnesota supercomputing institute research report UMSI*, 25:2005.
- Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183, Berlin, Germany. Association for Computational Linguistics.
- Martin Riedl. 2016. *Unsupervised Methods for Learning and Using Semantics of Natural Language*. Ph.D. thesis, Technische Universität Darmstadt, Darmstadt.
- Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China. Association for Computational Linguistics.
- Eugen Ruppert, Manuel Kaufmann, Martin Riedl, and Chris Biemann. 2015. Jobimviz: A web-based visualization for graph-based distributional semantic models. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 103–108, Beijing, China. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.
- Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.
- Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING’2014)*, pages 151–160, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.
- Tim Van de Cruys. 2010. Mining for meaning: The extraction of lexicosemantic knowledge from text. *Groningen Dissertations in Linguistics*, 82.
- Stijn Van Dongen. 2008. Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications*, 30(1):121–141.

- Alfredo Vellido, José David Martín, Fabrice Rossi, and Paulo J.G. Lisboa. 2011. Seeing is believing: The importance of visualization in real-world machine learning applications. In *Proceedings of the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'2011)*, pages 219–226, Bruges, Belgium.
- Alfredo Vellido, José D. Martín-Guerrero, and Paulo J.G. Lisboa. 2012. Making machine learning models interpretable. In *20th European Symposium on Artificial Neural Networks, ESANN*, volume 12, pages 163–172, Bruges, Belgium.
- Jean Véronis. 2004. HyperLex: Lexical cartography for information retrieval. *Computer Speech and Language*, 18:223–252.
- Heng Low Wee. 2010. Word Sense Prediction Using Decision Trees. Technical report, Department of Computer Science, National University of Singapore.
- Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'2002)*, pages 1–7, Taipei, Taiwan. Association for Computational Linguistics.
- Deniz Yuret. 2012. FASTSUBS: An efficient and exact procedure for finding the most likely lexical substitutes based on an n-gram language model. *IEEE Signal Processing Letters*, 19(11):725–728.
- Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden. Association for Computational Linguistics.

Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison

Alessandro Raganato, Jose Camacho-Collados and Roberto Navigli

Department of Computer Science

Sapienza University of Rome

{raganato, collados, navigli}@di.uniroma1.it

Abstract

Word Sense Disambiguation is a long-standing task in Natural Language Processing, lying at the core of human language understanding. However, the evaluation of automatic systems has been problematic, mainly due to the lack of a reliable evaluation framework. In this paper we develop a unified evaluation framework and analyze the performance of various Word Sense Disambiguation systems in a fair setup. The results show that supervised systems clearly outperform knowledge-based models. Among the supervised systems, a linear classifier trained on conventional local features still proves to be a hard baseline to beat. Nonetheless, recent approaches exploiting neural networks on unlabeled corpora achieve promising results, surpassing this hard baseline in most test sets.

1 Introduction

Word Sense Disambiguation (WSD) has been a long-standing task in Natural Language Processing (NLP). It lies at the core of language understanding and has already been studied from many different angles (Navigli, 2009; Navigli, 2012). However, the field seems to be slowing down due to the lack of groundbreaking improvements and the difficulty of integrating current WSD systems into downstream NLP applications (de Laccalle and Agirre, 2015). In general the field does not have a clear path, partially owing to the fact that identifying real improvements over existing approaches becomes a hard task with current evaluation benchmarks. This is mainly due to the lack of a unified framework, which prevents direct and fair comparison among systems. Even

though many evaluation datasets have been constructed for the task (Edmonds and Cotton, 2001; Snyder and Palmer, 2004; Navigli et al., 2007; Pradhan et al., 2007; Agirre et al., 2010a; Navigli et al., 2013; Moro and Navigli, 2015, *inter alia*), they tend to differ in format, construction guidelines and underlying sense inventory. In the case of the datasets annotated using WordNet (Miller, 1995), the *de facto* sense inventory for WSD, we encounter the additional barrier of having text annotated with different versions. These divergences are in the main solved individually by using or constructing automatic mappings. The quality check of such mapping, however, tends to be impractical and this leads to mapping errors which give rise to additional system inconsistencies in the experimental setting. This issue is directly extensible to the training corpora used by supervised systems. In fact, results obtained by supervised or semi-supervised systems reported in the literature are not completely reliable, because the systems may not necessarily have been trained on the same corpus, or the corpus was preprocessed differently, or annotated with a sense inventory different from the test data. Thus, together, the foregoing issues prevent us from drawing reliable conclusions on different models, as in some cases ostensible improvements may have been obtained as a consequence of the nature of the training corpus, the preprocessing pipeline or the version of the underlying sense inventory, rather than of the model itself. Moreover, because of these divergences, current systems tend to report results on a few datasets only, making it hard to perform a direct quantitative confrontation.

This paper offers two main contributions. First, we provide a complete evaluation framework for all-words Word Sense Disambiguation overcoming all the aforementioned limitations by (1) standardizing the WSD datasets and training corpora

into a unified format, (2) semi-automatically converting annotations from any dataset to WordNet 3.0, and (3) preprocessing the datasets by consistently using the same pipeline. Second, we use this evaluation framework to perform a fair quantitative and qualitative empirical comparison of the main techniques proposed in the WSD literature, including the latest advances based on neural networks.

2 State of the Art

The task of Word Sense Disambiguation consists of associating words in context with the most suitable entry in a pre-defined sense inventory. Depending on their nature, WSD systems are divided into two main groups: supervised and knowledge-based. In what follows we summarize the current state of these two types of approach.

2.1 Supervised WSD

Supervised models train different features extracted from manually sense-annotated corpora. These features have been mostly based on the information provided by the surroundings words of the target word (Keok and Ng, 2002; Navigli, 2009) and its collocations. Recently, more complex features based on word embeddings trained on unlabeled corpora have also been explored (Taghipour and Ng, 2015b; Rothe and Schütze, 2015; Iacobacci et al., 2016). These features are generally taken as input to train a linear classifier (Zhong and Ng, 2010; Shen et al., 2013). In addition to these conventional approaches, the latest developments in neural language models have motivated some researchers to include them in their WSD architectures (Kågebäck and Salomonsson, 2016; Melamud et al., 2016; Yuan et al., 2016).

Supervised models have traditionally been able to outperform knowledge-based systems (Navigli, 2009). However, obtaining sense-annotated corpora is highly expensive, and in many cases such corpora are not available for specific domains. This is the reason why some of these supervised methods have started to rely on unlabeled corpora as well. These approaches, which are often classified as *semi-supervised*, are targeted at overcoming the knowledge acquisition bottleneck of conventional supervised models (Pilehvar and Navigli, 2014). In fact, there is a line of research specifically aimed at automatically obtaining large amounts of high-quality sense-annotated corpora

(Taghipour and Ng, 2015a; Raganato et al., 2016; Camacho-Collados et al., 2016a).

In this work we compare supervised systems and study the role of their underlying sense-annotated training corpus. Since semi-supervised models have been shown to outperform fully supervised systems in some settings (Taghipour and Ng, 2015b; Başkaya and Jurgens, 2016; Iacobacci et al., 2016; Yuan et al., 2016), we evaluate and compare models using both manually-curated and automatically-constructed sense-annotated corpora for training.

2.2 Knowledge-based WSD

In contrast to supervised systems, knowledge-based WSD techniques do not require any sense-annotated corpus. Instead, these approaches rely on the structure or content of manually-curated knowledge resources for disambiguation. One of the first approaches of this kind was Lesk (1986), which in its original version consisted of calculating the overlap between the context of the target word and its definitions as given by the sense inventory. Based on the same principle, various works have adapted the original algorithm by also taking into account definitions from related words (Banerjee and Pedersen, 2003), or by calculating the distributional similarity between definitions and the context of the target word (Basile et al., 2014; Chen et al., 2014). Distributional similarity has also been exploited in different settings in various works (Miller et al., 2012; Camacho-Collados et al., 2015; Camacho-Collados et al., 2016b). In addition to these approaches based on distributional similarity, an important branch of knowledge-based systems found their techniques on the structural properties of semantic graphs from lexical resources (Agirre and Soroa, 2009; Guo and Diab, 2010; Ponzetto and Navigli, 2010; Agirre et al., 2014; Moro et al., 2014; Weissenborn et al., 2015; Tripodi and Pelillo, 2016). Generally, these graph-based WSD systems first create a graph representation of the input text and then exploit different graph-based algorithms over the given representation (e.g., PageRank) to perform WSD.

3 Standardization of WSD datasets

In this section we explain our pipeline for transforming any given evaluation dataset or sense-annotated corpus into a preprocessed unified for-

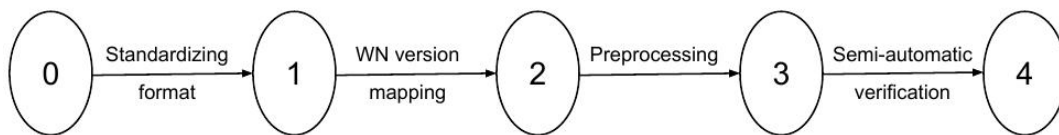


Figure 1: Pipeline for standardizing any given WSD dataset.

mat. In our pipeline we do not make any distinction between evaluation datasets and sense-annotated training corpora, as the pipeline can be applied equally to both types. For simplicity we will refer to both evaluation datasets and training corpora as WSD datasets.

Figure 1 summarizes our pipeline to standardize a WSD dataset. The process consists of four steps:

1. Most WSD datasets in the literature use a similar XML format, but they have some divergences on how to encode the information. For instance, the SemEval-15 dataset (Moro and Navigli, 2015) was developed for both WSD and Entity Linking and its format was especially designed for this latter task. Therefore, we decided to convert all datasets to a unified format. As unified format we use the XML scheme used for the SemEval-13 all-words WSD task (Navigli et al., 2013), where preprocessing information of a given corpus is also encoded.
2. Once the dataset is converted to a unified format, we map the sense annotations from its original WordNet version to 3.0, which is the latest version of WordNet used in evaluation datasets. This mapping is carried out semi-automatically. First, we use automatically-constructed WordNet mappings¹ (Daude et al., 2003). These mappings provide confidence values which we use to initially map senses whose mapping confidence is 100%. Then, the annotations of the remaining senses are manually checked, and re-annotated or removed whenever necessary². Additionally, in this step we decided to remove all annotations of auxiliary verbs, following the annotation guidelines of the latest WSD datasets.
3. The third step consists of preprocessing the given dataset. We used the Stanford

¹<http://nlp.lsi.upc.edu/tools/download-map.php>

²This manual correction involved less than 10% of all instances for the datasets for which this step was performed.

CoreNLP toolkit (Manning et al., 2014) for Part-of-Speech (PoS) tagging³ and lemmatization. This step is performed in order to ensure that all systems use the same preprocessed data.

4. Finally, we developed a script to check that the final dataset conforms to the aforementioned guidelines. In this final verification we also ensured that the sense annotations match the lemma and the PoS tag provided by Stanford CoreNLP by automatically fixing all divergences.

4 Data

In this section we summarize the WSD datasets used in the evaluation framework. To all these datasets we apply the standardization pipeline described in Section 3. First, we enumerate all the datasets used for the evaluation (Section 4.1). Second, we describe the sense-annotated corpora used for training (Section 4.2). Finally, we show some relevant statistics extracted from these resources (Section 4.3).

4.1 WSD evaluation datasets

For our evaluation framework we considered five standard all-words fine-grained WSD datasets from the Senseval and SemEval competitions:

- **Senseval-2** (Edmonds and Cotton, 2001). This dataset was originally annotated with WordNet 1.7. After standardization, it consists of 2282 sense annotations, including nouns, verbs, adverbs and adjectives.
- **Senseval-3 task 1** (Snyder and Palmer, 2004). The WordNet version of this dataset was 1.7.1. It consists of three documents from three different domains (editorial, news story and fiction), totaling 1850 sense annotations.

³In order to have a standard format which may be used by languages other than English, we provide coarse-grained PoS tags as given by the universal PoS tagset (Petrov et al., 2011).

	#Docs	#Sents	#Tokens	#Annotations	#Sense types	#Word types	Ambiguity
Senseval-2	3	242	5,766	2,282	1,335	1,093	5.4
Senseval-3	3	352	5,541	1,850	1,167	977	6.8
SemEval-07	3	135	3,201	455	375	330	8.5
SemEval-13	13	306	8,391	1,644	827	751	4.9
SemEval-15	4	138	2,604	1,022	659	512	5.5
SemCor	352	37,176	802,443	226,036	33,362	22,436	6.8
OMSTI	-	813,798	30,441,386	911,134	3,730	1,149	8.9

Table 1: Statistics of the WSD datasets used in the evaluation framework (after standardization).

- **SemEval-07 task 17** (Pradhan et al., 2007). This is the smallest among the five datasets, containing 455 sense annotations for nouns and verbs only. It was originally annotated using WordNet 2.1 sense inventory.
- **SemEval-13 task 12** (Navigli et al., 2013). This dataset includes thirteen documents from various domains. In this case the original sense inventory was WordNet 3.0, which is the same as the one that we use for all datasets. The number of sense annotations is 1644, although only nouns are considered.
- **SemEval-15 task 13** (Moro and Navigli, 2015). This is the most recent WSD dataset available to date, annotated with WordNet 3.0. It consists of 1022 sense annotations in four documents coming from three heterogeneous domains: biomedical, mathematics/computing and social issues.
- **OMSTI** (Taghipour and Ng, 2015a). OMSTI (*One Million Sense-Tagged Instances*) is a large corpus annotated with senses from the WordNet 3.0 inventory. It was automatically constructed by using an alignment-based WSD approach (Chan and Ng, 2005) on a large English-Chinese parallel corpus (Eisele and Chen, 2010, MultiUN corpus). OMSTI⁵ has already shown its potential as a training corpus by improving the performance of supervised systems which add it to existing training data (Taghipour and Ng, 2015a; Iacobacci et al., 2016).

4.2 Sense-annotated training corpora

We now describe the two WordNet sense-annotated corpora used for training the supervised systems in our evaluation framework:

- **SemCor** (Miller et al., 1994). SemCor⁴ is a manually sense-annotated corpus divided into 352 documents for a total of 226,040 sense annotations. It was originally tagged with senses from the WordNet 1.4 sense inventory. SemCor is, to our knowledge, the largest corpus manually annotated with WordNet senses, and is the main corpus used in the literature to train supervised WSD systems (Agirre et al., 2010b; Zhong and Ng, 2010).

⁴We downloaded the SemCor 3.0 version at web.eecs.umich.edu/~mihalcea/downloads.html

4.3 Statistics

Table 1 shows some statistics⁶ of the WSD datasets and training corpora which we use in the evaluation framework. The number of sense annotations varies across datasets, ranging from 455 annotations in the SemEval-07 dataset, to 2,282 annotations in the Senseval-2 dataset. As regards sense-annotated corpora, OMSTI is made up of almost 1M sense annotations, a considerable increase over the number of sense annotations of SemCor. However, SemCor is much more balanced in terms of unique senses covered (3,730 covered by OMSTI in contrast to over 33K covered by SemCor). Additionally, while OMSTI was constructed automatically, SemCor was manually built and, hence, its quality is expected to be higher.

Finally, we calculated the ambiguity level of each dataset, computed as the total number of can-

⁵In this paper we refer to the portion of sense-annotated data from the MultiUN corpus as OMSTI. Note that OMSTI was released along with SemCor.

⁶Statistics included in Table 1: number of documents (#Docs), sentences (#Sents), tokens (#Tokens), sense annotations (#Annotations), sense types covered (#Sense types), annotated lemma types covered (#Word types), and ambiguity level (Ambiguity). There was no document information in the OMSTI data released by Taghipour and Ng (2015a).

didate senses (i.e., senses sharing the surface form of the target word) divided by the number of sense annotations. The highest ambiguity is found on OMSTI, which, despite being constructed automatically, contains a high coverage of ambiguous words. As far as the evaluation competition datasets are concerned, the ambiguity may give a hint as to how difficult a given dataset may be. In this case, SemEval-07 displays the highest ambiguity level among all evaluation datasets.

5 Evaluation

The evaluation framework consists of the WSD evaluation datasets described in Section 4.1. In this section we use this framework to perform an empirical comparison among a set of heterogeneous WSD systems. The systems used in the evaluation are described in detail in Section 5.1, the results are shown in Section 5.2 and a detailed analysis is presented in Section 5.3.

5.1 Comparison systems

We include three supervised (Section 5.1.1) and three knowledge-based (Section 5.1.2) all-words WSD systems in our empirical comparison.

5.1.1 Supervised

To ensure a fair comparison, all supervised systems use the same corpus for training: SemCor and Semcor+OMSTI⁷ (see Section 4.2). In the following we describe the three supervised WSD systems used in the evaluation:

- **IMS** (Zhong and Ng, 2010) uses a Support Vector Machine (SVM) classifier over a set of conventional WSD features. IMS⁸ is built on a flexible framework which allows an easy integration of different features. The default implementation includes surrounding words, PoS tags of surroundings words, and local collocations as features.
- **IMS+embeddings** (Taghipour and Ng, 2015b; Rothe and Schütze, 2015; Iacobacci et al., 2016). These approaches have shown the potential of using word embeddings on the WSD task. Iacobacci et al. (2016) carried

out a comparison of different strategies for integrating word embeddings as a feature in WSD. In this paper we consider the two best configurations in Iacobacci et al. (2016)⁹: using all IMS default features including and excluding surrounding words (IMS+emb and IMS_s+emb, respectively). In both cases word embeddings are integrated using exponential decay (i.e., word weights drop exponentially as the distance towards the target word increases). Likewise, we use Iacobacci et al.’s suggested learning strategy and hyperparameters to train the word embeddings: Skip-gram model of Word2Vec¹⁰ (Mikolov et al., 2013) with 400 dimensions, ten negative samples and a window size of ten words. As unlabeled corpus to train the word embeddings we use the English ukWaC corpus¹¹ (Baroni et al., 2009), which is made up of two billion words from paragraphs extracted from the web.

- **Context2Vec** (Melamud et al., 2016). Neural language models have recently shown their potential for the WSD task (Kågebäck and Salomonsson, 2016; Yuan et al., 2016). In this experiment we replicated the approach of Melamud et al. (2016, Context2Vec), for which the code¹² is publicly available. This approach is divided in three steps. First, a bidirectional LSTM recurrent neural network is trained on an unlabeled corpus (we considered the same ukWaC corpus used by the previous comparison system). Then, a context vector is learned for each sense annotation in the training corpus. Finally, the sense annotation whose context vector is closer to the target word’s context vector is selected as the intended sense.

Finally, as baseline we included the Most Frequent Sense (**MFS**) heuristic, which for each target word selects the sense occurring the highest number of times in the training corpus.

⁷As already noted by Taghipour and Ng (2015a), supervised systems trained on only OMSTI obtain lower results than when trained along with SemCor, mainly due to OMSTI’s lack of coverage in target word types.

⁸We used the original implementation available at <http://www.comp.nus.edu.sg/~nlp/software.html>

⁹We used the implementation available at https://github.com/iacobac/ims_wsd_emb

¹⁰code.google.com/archive/p/word2vec/

¹¹<http://wacky.sslmit.unibo.it/doku.php?id=corpora>

¹²<https://github.com/orenmel/context2vec>

5.1.2 Knowledge-based

In this section we describe the three knowledge-based WSD models used in our empirical comparison:

- **Lesk** (Lesk, 1986) is a simple knowledge-based WSD algorithm that bases its calculations on the overlap between the definitions of a given sense and the context of the target word. For our experiments we replicated the extended version of the original algorithm in which definitions of related senses are also considered and the conventional term frequency-inverse document frequency (Jones, 1972, *tf-idf*) is used for word weighting (Banerjee and Pedersen, 2003, Lesk_{ext}). Additionally, we included the enhanced version of Lesk in which word embeddings¹³ are leveraged to compute the similarity between definitions and the target context (Basile et al., 2014, Lesk_{ext+emb})¹⁴.
- **UKB** (Agirre and Soroa, 2009; Agirre et al., 2014) is a graph-based WSD system which makes use of random walks over a semantic network (WordNet graph in this case). UKB¹⁵ applies the Personalized Page Rank algorithm (Haveliwala, 2002) initialized using the context of the target word. Unlike most WSD systems, UKB does not back-off to the WordNet first sense heuristic and it is self-contained (i.e., it does not make use of any external resources/corpora). We used both default configurations from UKB: using the full WordNet graph (UKB) and the full graph including disambiguated glosses as connections as well (UKB_{gloss}).
- **Babelfy** (Moro et al., 2014) is a graph-based disambiguation approach which exploits random walks to determine connections between synsets. Specifically, Babelfy¹⁶ uses random walks with restart (Tong et al., 2006) over BabelNet (Navigli and Ponzetto, 2012), a large semantic network integrating WordNet among other resources such as Wikipedia

¹³We used the same word embeddings described in Section 5.1.1 for IMS+emb.

¹⁴We used the implementation from <https://github.com/pippokill/lesk-wsd-dsm>. In this implementation additional definitions from BabelNet are considered.

¹⁵We used the last implementation available at <http://ixa2.si.ehu.es/ukb/>

¹⁶We used the Java API from <http://babelfy.org>

or Wiktionary. Its algorithm is based on a densest subgraph heuristic for selecting high-coherence semantic interpretations of the input text. The best configuration of Babelfy takes into account not only the target sentence in which the target word occurs, but also the whole document.

As knowledge-based baseline we included the **WordNet first sense**. This baseline simply selects the candidate which is considered as first sense in WordNet 3.0. Even though the sense order was decided on the basis of semantically-tagged text, we considered it as knowledge-based in this experiment as this information is already available in WordNet. In fact, knowledge-based systems like Babelfy include this information in their pipeline. Despite its simplicity, this baseline has been shown to be hard to beat by automatic WSD systems (Navigli, 2009; Agirre et al., 2014).

5.2 Results

Table 2 shows the F-Measure performance of all comparison systems on the five all-words WSD datasets. Since not all test word instances are covered by the corresponding training corpora, supervised systems have a maximum F-Score (*ceiling* in the Table) they can achieve. Nevertheless, supervised systems consistently outperform knowledge-based systems across datasets, confirming the results of Pilehvar and Navigli (2014). A simple linear classifier over conventional WSD features (i.e., IMS) proves to be robust across datasets, consistently outperforming the MFS baseline. The recent integration of word embeddings as an additional feature is beneficial, especially as a replacement of the feature based on the surface form of surrounding words (i.e., IMS_{s+emb}). Moreover, recent advances on neural language models (in the case of Context2Vec a bi-directional LSTM) appear to be highly promising for the WSD task according to the results, as Context2Vec outperforms IMS in most datasets.

On the other hand, it is also interesting to note the performance inconsistencies of systems across datasets, as in all cases there is a large performance gap between the best and the worst performing dataset. As explained in Section 4.3, the ambiguity level may give a hint as to how difficult the corresponding dataset may be. In fact, WSD systems obtain relatively low results in SemEval-07, which is the most ambiguous dataset (see Table 1).

	Tr. Corpus	System	Senseval-2	Senseval-3	SemEval-07	SemEval-13	SemEval-15
Supervised	SemCor	IMS	70.9	69.3	61.3	65.3	69.5
		IMS+emb	71.0	69.3	60.9	67.3	71.3
		IMS _s +emb	72.2	70.4	62.6	65.9	71.5
		Context2Vec	71.8	69.1	61.3	65.6	71.9
		MFS	65.6	66.0	54.5	63.8	67.1
		<i>Ceiling</i>	<i>91.0</i>	<i>94.5</i>	<i>93.8</i>	<i>88.6</i>	<i>90.4</i>
	SemCor + OMSTI	IMS	72.8	69.2	60.0	65.0	69.3
		IMS+emb	70.8	68.9	58.5	66.3	69.7
		IMS _s +emb	73.3	69.6	61.1	66.7	70.4
		Context2Vec	72.3	68.2	61.5	67.2	71.7
		MFS	66.5	60.4	52.3	62.6	64.2
		<i>Ceiling</i>	<i>91.5</i>	<i>94.9</i>	<i>94.7</i>	<i>89.6</i>	<i>91.1</i>
Knowledge	-	Lesk _{ext}	50.6	44.5	32.0	53.6	51.0
		Lesk _{ext} +emb	63.0	63.7	56.7	66.2	64.6
		UKB	56.0	51.7	39.0	53.6	55.2
		UKB _{gloss}	60.6	54.1	42.0	59.0	61.2
		Babelfy	67.0	63.5	51.6	66.4	70.3
		WN 1 st sense	66.8	66.2	55.2	63.0	67.8

Table 2: F-Measure percentage of different models in five all-words WSD datasets.

	Nouns	Verbs	Adj.	Adv.	All
#Instances	4,300	1,652	955	346	7,253
Ambiguity	4.8	10.4	3.8	3.1	5.8

Table 3: Number of instances and ambiguity level of the concatenation of all five WSD datasets.

However, this is the dataset in which supervised systems achieve a larger margin with respect to the MFS baseline, which suggests that, in general, the MFS heuristic does not perform accurately on highly ambiguous words.

5.3 Analysis

To complement the results from the previous section, we additionally carried out a detailed analysis about the global performance of each system and divided by PoS tag. To this end, we concatenated all five datasets into a single dataset. This resulted in a large evaluation dataset of 7,253 instances to disambiguate (see Table 3). Table 4 shows the F-Measure performance of all comparison systems on the concatenation of all five WSD evaluation datasets, divided by PoS tag. IMS_s+emb trained on SemCor+OMSTI achieves the best overall results, slightly above Context2Vec trained on the same corpus. In what follows we describe some of the main findings extracted from our analysis.

Training corpus. In general, the results of supervised systems trained on SemCor only (manually-annotated) are lower than training

simultaneously on both SemCor and OMSTI (automatically-annotated). This is a promising finding, which confirms the results of previous works (Raganato et al., 2016; Iacobacci et al., 2016; Yuan et al., 2016) and encourages further research on developing reliable automatic or semi-automatic methods to obtain large amounts of sense-annotated corpora in order to overcome the knowledge-acquisition bottleneck. For instance, Context2Vec improves 0.4 points overall when adding the automatically sense-annotated OMSTI as part of the training corpus, suggesting that more data, even if not perfectly clean, may be beneficial for neural language models.

Knowledge-based vs. Supervised. One of the main conclusions that can be taken from the evaluation is that supervised systems clearly outperform knowledge-based models. This may be due to the fact that in many cases the main disambiguation clue is given by the immediate local context. This is particularly problematic for knowledge-based systems, as they take equally into account all the words within a sentence (or document in the case of Babelfy). For instance, in the following sentence, both UKB and Babelfy fail to predict the correct sense of *state*:

In sum, at both the federal and state government levels at least part of the seemingly irrational behavior voters display in the voting booth may have an exceedingly rational explanation.

	Tr. Corpus	System	Nouns	Verbs	Adjectives	Adverbs	All
Supervised	SemCor	IMS	70.4	56.1	75.6	82.9	68.4
		IMS+emb	71.8	55.4	76.1	82.7	69.1
		IMS _s +emb	71.9	56.9	75.9	84.7	69.6
		Context2Vec	71.0	57.6	75.2	82.7	69.0
		MFS	67.6	49.6	73.1	80.5	64.8
		<i>Ceiling</i>	<i>89.6</i>	<i>95.1</i>	<i>91.5</i>	<i>96.4</i>	<i>91.5</i>
	SemCor + OMSTI	IMS	70.5	56.9	76.8	82.9	68.8
		IMS+emb	71.0	53.3	77.1	82.7	68.3
		IMS _s +emb	72.0	56.5	76.6	84.7	69.7
		Context2Vec	71.7	55.8	77.2	82.7	69.4
		MFS	65.8	45.9	72.7	80.5	62.9
		<i>Ceiling</i>	<i>90.4</i>	<i>95.8</i>	<i>91.8</i>	<i>96.4</i>	<i>92.1</i>
Knowledge	-	Lesk _{ext}	54.1	27.9	54.6	60.3	48.7
		Lesk _{ext} +emb	69.8	51.2	51.7	80.6	63.7
		UKB	56.7	39.3	63.9	44.0	53.2
		UKB _{gloss}	62.1	38.3	66.8	66.2	57.5
		Babelify	68.6	49.9	73.2	79.8	65.5
		WN 1 st sense	67.6	50.3	74.3	80.9	65.2

Table 4: F-Measure percentage of different models on the concatenation of all five WSD datasets.

In this sentence, *state* is annotated with its *administrative districts of a nation* sense in the gold standard. The main disambiguation clue seems to be given by its previous and immediate subsequent words (*federal* and *government*), which tend to co-occur with this particular sense. However, knowledge-based WSD systems like UKB or Babelify give the same weight to all words in context, underrating the importance of this local disambiguation clue in the example. For instance, UKB disambiguates *state* with the sense defined as *the way something is with respect to its main attributes*, probably biased by words which are not immediately next to the target word within the sentence, e.g., *irrational*, *behaviour*, *rational* or *explanation*.

Low overall performance on verbs. As can be seen from Table 4, the F-Measure performance of all systems on verbs is in all cases below 58%. This can be explained by the high granularity of verbs in WordNet. For instance, the verb *keep* consists of 22 different meanings in WordNet 3.0, six of them denoting “possession and transfer of possession”¹⁷. In fact, the average ambiguity level of all verbs in this evaluation framework is 10.4 (see

¹⁷<https://wordnet.princeton.edu/man/lexnames.5WN.html>

Table 3), considerably greater than the ambiguity on other PoS tags, e.g., 4.8 in nouns. Nonetheless, supervised systems manage to comfortably outperform the MFS baseline, which does not seem to be reliable for verbs given their high ambiguity.

Influence of preprocessing. As mentioned in Section 3, our evaluation framework provides a preprocessing of the corpora with Stanford CoreNLP. This ensures a fair comparison among all systems but may introduce some annotation inaccuracies, such as erroneous PoS tags. However, for English these errors are minimal¹⁸. For instance, the global error rate of the Stanford PoS tagger in all disambiguation instances is 3.9%, which were fixed as explained in Section 3.

Bias towards the Most Frequent Sense. After carrying out an analysis on the influence of MFS in WSD systems¹⁹, we found that all supervised systems suffer a strong bias towards the MFS, with all IMS-based systems disambiguating over 75% of instances with their MFS. Context2Vec is slightly less affected by this bias, with 71.5% (SemCor) and 74.7% (SemCor+OMSTI) of answers corre-

¹⁸Even if preprocessing plays a minimal role for English, it may be of higher importance for other languages, e.g., morphologically richer languages (Eger et al., 2016).

¹⁹See Postma et al. (2016) for an interesting discussion on the bias of current WSD systems towards the MFS.

sponding to the MFS. Interestingly, this MFS bias is also present in graph knowledge-based systems. In fact, Calvo and Gelbukh (2015) had already shown how the MFS correlates strongly with the number of connections in WordNet.

Knowledge-based systems. For knowledge-based systems the WN first sense baseline proves still to be extremely hard to beat. The only knowledge-based system that overall manages to beat this baseline is Babelfy, which, in fact, uses information about the first sense in its pipeline. Babelfy’s default pipeline includes a confidence threshold in order to decide whether to disambiguate or back-off to the first sense. In total, Babelfy backs-off to WN first sense in 63% of all instances. Nonetheless, it is interesting to note the high performance of Babelfy and Lesk_{ext}+emb on noun instances (outperforming the first sense baseline by 1.0 and 2.2 points, respectively) in contrast to their relatively lower performance on verbs, adjectives²⁰ and adverbs. We believe that this is due to the nature of the lexical resource used by these two systems, i.e., BabelNet. BabelNet includes Wikipedia as one of its main sources of information. However, while Wikipedia provides a large amount of semantic connections and definitions for nouns, this is not the case for verbs, adjectives and adverbs, as they are not included in Wikipedia and their source of information mostly comes from WordNet only.

6 Conclusion and Future Work

In this paper we presented a unified evaluation framework for all-words WSD. This framework is based on evaluation datasets taken from Senseval and SemEval competitions, as well as manually and automatically sense-annotated corpora. In this evaluation framework all datasets share a common format, sense inventory (i.e., WordNet 3.0) and preprocessing pipeline, which eases the task of researchers to evaluate their models and, more importantly, ensures a fair comparison among all systems. The whole evaluation framework²¹, including guidelines for researchers to include their own sense-annotated datasets and a script to validate their conformity to the guidelines, is available at <http://lcl.uniroma1.it/wsdeval>.



²⁰The poor performance of Lesk_{ext}+emb on adjective instances is particularly noticeable.

²¹We have additionally set up a CodaLab competition based on this evaluation framework.

We used this framework to perform an empirical comparison among a set of heterogeneous WSD systems, including both knowledge-based and supervised ones. Supervised systems based on neural networks achieve the most promising results. Given our analysis, we foresee two potential research avenues focused on semi-supervised learning: (1) exploiting large amounts of unlabeled corpora for learning word embeddings or training neural language models, and (2) automatically constructing high-quality sense-annotated corpora to be used by supervised WSD systems. As far as knowledge-based systems are concerned, enriching knowledge resources with semantic connections for non-nominal mentions may be an important step towards improving their performance.

For future work we plan to further extend our unified framework to languages other than English, including SemEval multilingual WSD datasets, as well as to other sense inventories such as Open Multilingual WordNet, BabelNet and Wikipedia, which are available in different languages.

Acknowledgments

 The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234. 

Jose Camacho-Collados is supported by a Google PhD Fellowship in Natural Language Processing.

References

- Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of EACL*, pages 33–41.
- Eneko Agirre, Oier Lopez De Lacalle, Christiane Fellbaum, Andrea Marchetti, Antonio Toral, and Piek Vossen. 2010a. Semeval-2010 task 17: All-words word sense disambiguation on a specific domain. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 123–128.
- Eneko Agirre, Oier Lopez De Lacalle, Christiane Fellbaum, Andrea Marchetti, Antonio Toral, and Piek Vossen. 2010b. Semeval-2010 task 17: All-words word sense disambiguation on a specific domain. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 123–128.
- Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word

- sense disambiguation. *Computational Linguistics*, 40(1):57–84.
- Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlap as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 805–810, Acapulco, Mexico.
- Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.
- Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1591–1600, Dublin, Ireland.
- Osman Başkaya and David Jurgens. 2016. Semi-supervised learning with induced word senses for state of the art word sense disambiguation. *Journal of Artificial Intelligence Research*, 55:1025–1058.
- Hiram Calvo and Alexander Gelbukh. 2015. Is the most frequent sense of a word better connected in a semantic network? In *International Conference on Intelligent Computing*, pages 491–499. Springer.
- José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A Unified Multilingual Semantic Representation of Concepts. In *Proceedings of ACL*, pages 741–751.
- José Camacho-Collados, Claudio Delli Bovi, Alessandro Raganato, and Roberto Navigli. 2016a. A Large-Scale Multilingual Disambiguation of Glosses. In *Proceedings of LREC*, pages 1701–1708, Portoroz, Slovenia.
- José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016b. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.
- Yee Seng Chan and Hwee Tou Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *AAAI*, volume 5, pages 1037–1042.
- Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*, pages 1025–1035, Doha, Qatar.
- Jordi Daude, Lluís Padro, and German Rigau. 2003. Validation and tuning of wordnet mapping techniques. In *Proceedings of RANLP*.
- Oier Lopez de Lacalle and Eneko Agirre. 2015. A methodology for word sense disambiguation at 90% based on large-scale crowdsourcing. *Lexical and Computational Semantics (*SEM 2015)*, page 61.
- Philip Edmonds and Scott Cotton. 2001. Senseval-2: Overview. In *Proceedings of The Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–6, Toulouse, France.
- Steffen Eger, Rüdiger Gleim, and Alexander Mehler. 2016. Lemmatization and morphological tagging in german and latin: A comparison and a survey of the state-of-the-art. In *Proceedings of LREC 2016*.
- Andreas Eisele and Yu Chen. 2010. MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872.
- Weiwei Guo and Mona T. Diab. 2010. Combining orthogonal monolingual and multilingual sources of evidence for all words WSD. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1542–1551, Uppsala, Sweden.
- Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web*, pages 517–526, Hawaii, USA.
- Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of ACL*, pages 897–907, Berlin, Germany.
- Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.
- Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional lstm. *arXiv preprint arXiv:1606.03568*.
- L. Y. Keok and H. T. Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the 7th Conference on Empirical Methods in Natural Language Processing*, pages 41–48, Philadelphia, USA.
- Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual Conference on Systems Documentation*, Toronto, Ontario, Canada, pages 24–26.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of CONLL*.

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. 1994. Using a semantic concordance for sense identification. In *Proceedings of the workshop on Human Language Technology*, pages 240–243. Association for Computational Linguistics.
- Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *COLING*, pages 1781–1796.
- George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. *Proceedings of SemEval-2015*.
- Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2:231–244.
- Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.
- Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. SemEval-2007 task 07: Coarse-grained English all-words task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, pages 30–35.
- Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Proceedings of SemEval 2013*, pages 222–231.
- Roberto Navigli. 2009. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.
- Roberto Navigli. 2012. A quick tour of word sense disambiguation, induction and related approaches. In *SOFSEM 2012: Theory and practice of computer science*, pages 115–129. Springer.
- Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.
- Mohammad Taher Pilehvar and Roberto Navigli. 2014. A large-scale pseudoword-based evaluation framework for state-of-the-art Word Sense Disambiguation. *Computational Linguistics*, 40(4).
- Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich Word Sense Disambiguation rivaling supervised system. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1522–1531, Uppsala, Sweden.
- Marten Postma, Ruben Izquierdo, Eneko Agirre, German Rigau, and Piek Vossen. 2016. Addressing the MFS Bias in WSD systems. In *Proceedings of LREC*, Portoroz, Slovenia.
- Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of SemEval*, pages 87–92.
- Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2016. Automatic Construction and Evaluation of a Large Semantically Enriched Wikipedia. In *Proceedings of IJCAI*, pages 2894–2900, New York City, NY, USA, July.
- Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*, pages 1793–1803, Beijing, China.
- Hui Shen, Razvan Bunescu, and Rada Mihalcea. 2013. Coarse to fine grained sense disambiguation in wikipedia. *Proc. of *SEM*, pages 22–31.
- Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, Barcelona, Spain, pages 41–43, Barcelona, Spain.
- Kaveh Taghipour and Hwee Tou Ng. 2015a. One million sense-tagged instances for word sense disambiguation and induction. *CoNLL 2015*, page 338.
- Kaveh Taghipour and Hwee Tou Ng. 2015b. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. *Proceedings of NAACL HLT 2015*, pages 314–323.
- Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*, pages 613–622.
- Rocco Tripodi and Marcello Pelillo. 2016. A game-theoretic approach to word sense disambiguation. *arXiv preprint arXiv:1606.07711*.
- Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-Objective Optimization for the Joint Disambiguation of Nouns and Named Entities. In *Proceedings of ACL*, pages 596–605, Beijing, China.
- Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING*, pages 1374–1385.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense:
A wide-coverage Word Sense Disambiguation system for free text. In *Proceedings of the ACL System Demonstrations*, pages 78–83.

Which is the Effective Way for Gaokao: Information Retrieval or Neural Networks?

Shangmin Guo^{†1}, Xiangrong Zeng^{†1,2}, Shizhu He¹,
Kang Liu¹ and Jun Zhao¹

¹National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China

²University of Chinese Academy of Sciences, Beijing, 10049, China
{shangmin.guo, xiangrong.zeng}@nlpr.ia.ac.cn
{shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

As one of the most important test of China, Gaokao is designed to be difficult enough to distinguish the excellent high school students. In this work, we detailed the Gaokao History Multiple Choice Questions(GKHMC) and proposed two different approaches to address them using various resources. One approach is based on entity search technique (IR approach), the other is based on text entailment approach where we specifically employ deep neural networks(NN approach). The result of experiment on our collected real Gaokao questions showed that they are good at different categories of questions, i.e. IR approach performs much better at entity questions(EQs) while NN approach shows its advantage on sentence questions(SQs). Our new method achieves state-of-the-art performance and show that it's indispensable to apply hybrid method when participating in the real-world tests.

1 Introduction

Gaokao, namely the National College Entrance Examination, is the most important examination for Chinese senior high school students. Every college in China, no matter it is Top10 or Top100, would only accept the exam-takers whose Gaokao score is higher than its threshold score. As there are almost 10 million students take the examination every year, Gaokao needs to be difficult enough to distinguish the excellent students. Therefore, it includes various types of questions such as multiple-choice questions, short-answer

[†] Both of the two authors contributed equally to this paper.

After the World War II, U.S. and Soviet Union are fighting against each other in politics, economics and military. To promote the development of economics in Socialist Countries, Soviet Union establish The Council for Mutual Economic Assistance. This is against	
A. Truman Doctrine	B. Marshall Plan
C. NATO	D. Federal Republic of Germany

Entity Question

From Qin and Han Dynasties to Ming Dynasty, businessmen are always at the bottom of hierarchy. One reason for this is that the ruling class thought the businessmen	
A. are not engaged in production	B. do not respect Confucianism
C. do not respect the clan	D. do not pay tax

Sentence Question

Figure 1: Examples of questions and their types. The upper one is an entity question. The lower one is a sentence question.

questions and essays and it covers several different subjects, like Chinese, Math, History and etc. In this work, we focus on Gaokao History Multiple Choice questions which is denoted as GKHMC. Both of the factoid question answering task and reading comprehension task are similar to GKHMC. But, the GKHMC questions have their own characteristics.

A multiple-choice question in GKHMC such as the examples shown in Figure 1 is composed of a question stem and four candidates. Our goal is to figure out the only one correct candidate. But, there are certain obstacles to achieve it. First, several background sentences and a lead-in sentence conjointly constitutes the question stem, which makes these questions more complicated than former one-sentence-long factoid questions that can be handled by the existing approaches, like (Kolomiyet and Moens, 2011; Kwiatkowski et al., 2013; Berant and Liang, 2014; Yih et al., 2015). Secondly, the background sentences generally contain various clues to figure out the historical events or personages which may be the perdue key to answer the question. These clues may include Tang poem and Song iambic verse, domain-specific expressions, even some mixture of mod-

Question Type	Candidate Type	Count
EQ	Entities	160
SQ	Sentence	584
ALL	Whatever	744

Table 1: The GKHMC dataset.

ern Chinese and excerpt from ancient books and etc. The dependence of background knowledge makes the models that are designed for reading comprehension such as (Peñas et al., 2013; Richardson et al., 2013) fail. Thirdly, the diversity of candidates’ granularity, i.e. candidates can either be entities or sentences, makes it harder to match the candidate and stem. So, the answer selection is disparate from the former approaches whose candidates are usually just entities. Lastly, as the candidates are already given, the answer generation step in former neural network approaches based question answering system is no longer necessary.

As mentioned above and shown in Figure 1, in accordance with candidates’ granularity, the GKHMC questions can be divided into two types: entity questions(EQs) and sentence questions(SQs). Entity questions are those whose candidates are all entities, no matter they are people, dynasties, warfares or something else. And, sentence questions are those whose candidates are all sentences. We observe that such two types of questions have their own specific characteristics. Most of background sentences in EQs are description of the right candidate, so it may be particularly suitable to apply information retrieval like approach to handle them. Meanwhile, as the background sentences and lead-in sentences in SQs are more like the entailing text, these questions aren’t appropriate to be addressed by lexically searching and matching. Therefore, it seems that it’s more resonable to resolve SQs by using textual reasoning techniques.

In this paper, we wonder about which kind of approach is more effective for GKHMC. Furthermore, whether we should select specific method to work out different types of questions. In terms of various characteristics of GKHMC questions, we introduce two independent approaches to address them. One is based on entity search technique (IR approach) and the other is based on a text entailment approach where we specifically employ deep neural networks (NN approach). In IR approach, we use the key entities and relationships extracted

from questions to form a query, then inquire this query in all the text resources to get the most relevant candidate. In NN approach, we take the question text and every candidate to form four statements respectively, then judge how possible every statement is right so that we can figure out which is most likely to be the correct answer.

To test the two approaches’ performance, we collected and classified the multiple-choice questions in Gaokao test papers from 2011 to 2015 all over the country, and they are released. From the result, we find that the performance of two approaches are significantly discrepant at each kind of questions. That is, IR approach shows noticeable advantages on EQs, while NN approach performs much better on SQs. This will be further discussed in Section 4.4.

In this paper, our contributions are as follows:

- We gave a detailed description of the Gaokao History Multiple Choice Questions task and showed its importance and difficulty.
- We released a dataset¹ for this task. The dataset is manually collected and classified. All questions in the dataset are real Gaokao quesitons from 2011 to 2015.
- We introduced two different approaches for this task. Each approach achieved a promising results. We also compared this two approaches and found that they are complementary, i.e. they are good at different types of questions.
- We introduced permanent provisional memory network(PPMN) to model the joint background knowledge and sentences in question stem, and it beats existing memory networks on SQs.

2 Dataset

As described in the Introduction, we collected the historical multiple-choice questions from Gaokao all over the country in rencent five years. However, quite a lot contain graphs or tables which require the techniques beyond natural language processing(NLP). So, we filter out this part of questions and manually classified the left into two parts: EQs and SQs. The number of different kinds of questions are listed in Table 1. The examples of

¹ <https://github.com/IACASNLPIR/GKHMC/tree/master/data>

different types of questions translated into English are shown in Figure 1.

It is worth mentioning that there is a special type of questions on test papers named sequential questions. The candidates of this kind of questions are just some ordered numbers. Every number stands for a certain content which is given in question stem. We simply replace every sequential number in candidates with their corresponding contents. Then, we can classify these questions as EQs or SQs according to the type of contents.

We also collected a wide diversity of resources including Baidu Encyclopedia, textbooks and practice questions as our external knowledge when inquiring the generated query. Baidu Encyclopedia which is also known as Baidu Baike, is something like Wikipedia, but the content of it is written in Chinese. We denote this resource as BAIKE. The textbooks resource contains three compulsory history textbooks published by People’s Education Press. We denote them as BOOK. And we gathered about 50,000 practice questions and their answers, and this is denoted as TIKU.

3 Approaches²

3.1 IR Approach

The GKHMC questions require figuring out the most relevant candidate to the question stem from the four given candidates. Our IR approach is inspired by this observation. The diagram of IR approach is illustrated in Figure 2.

The pipeline of IR approach is: (1) use the classifier to automatically classify the question and select the weights according to the classification result; (2) calculate the relevance scores for every candidate (we introduce three different methods with seven score functions to calculate the relevance scores) and combine them together with specific weights; (3) choose the candidate with highest score as right answer. Despite the simplicity of it, IR approach achieves a promising result in experiment.

3.1.1 Naive Bayes Classifier

We build a naive bayes classifier to classify questions. Using length of candidates, entity number of candidates and verb number of candidates as features, every question is classified as EQ or SQ. When building the classifier, we do 10-folder cross

² The codes of this project can be obtained at <https://github.com/IACASNLPIR/GKHMC>

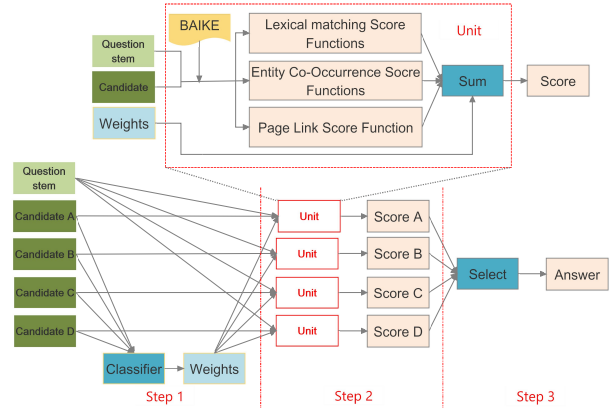


Figure 2: Pipeline of IR approach.

validation on the GKHMC dataset and the results are 90.00% precision and 84.38% recall in EQs and 95.79% precision and 97.43% recalls in SQs.

3.1.2 Score Functions

To calculate the relevance between question stem and candidates, we introduce three different methods with seven score functions, which are summarized in Table 2.

Lexical Matching Score: Since the correct candidate usually directly related to question stem, it’s reasonable to assume that the facts in question stem may appear in documents related to them, together with the correct candidate. Here we introduce our lexical matching score functions, taking BAIKE as our external resource. The four queries are formed by each candidate and question stem separately. Then we retrieval every query and sum up the scores of the top three returned documents as the lexical matching score. We use $score_{top_i}$ to denote the score of the top i -th returned documents. $score_{top_i}$ is calculated by Lucene’s TFIDFSimilarity function³. The lexical matching score $Score_{lexical}(candidate_k)$ is calculated as

$$Score_{lexical}(candidate_k) = \sum_{i=1}^3 (score_{top_i}). \quad (1)$$

We build indices for BAIKE with different grains. The index built for every BAIKE document is denoted as BAIKE Document Index(BDI). The index built for every paragraph in BAIKE is denoted as BAIKE Paragraph Index(BPI). And, the index built for every sentence in BAIKE is called BAIKE Sentence Index(BSI).

³ <https://lucene.apache.org/core/>

We denote the lexical matching score function using BDI, BPI and BSI as $Score_{BDI}$, $Score_{BPI}$ and $Score_{BSI}$ respectively.

Entity Co-Occurrence Score: We also consider the relevance of entities in co-occurrence aspect. If two entities often appearing together, we assume that they are relevant. We use normalized google distance (Cilibrasi and Vitanyi, 2007) to calculate the entity co-occurrence score $Score_{co}(candidate_k)$.

$$NGD(e_i, e_j) = \frac{Max(e_i, e_j) - \log f(e_i, e_j)}{\log N - Min(e_i, e_j)} \quad (2)$$

$$Max(e_i, e_j) = \max\{\log f(e_i), \log f(e_j)\} \quad (3)$$

$$Min(e_i, e_j) = \min\{\log f(e_i), \log f(e_j)\} \quad (4)$$

$$Score_{co}(candidate_k) = -\log(NGD(e_i, e_j)) \quad (5)$$

where

$$e_i \in E_{stem}, e_j \in E_{candidate_k}.$$

In which, e_i is entity; $f(e_i)$ is the number of parts which contain entity e_i ; $f(e_i, e_j)$ is the number of parts which contain both entity e_i and e_j ; E_{stem} and $E_{candidate_k}$ denotes the entities in question stem and candidate.

The entity co-occurrence could be in document, paragraph or sentence, and they are denoted as $Score_{BDC}$, $Score_{BPC}$ and $Score_{BSC}$ respectively.

Page Link Score: Inspired from PageRank algorithm (Page et al., 1999), we assume that entities have links to each other are relevant. Here we introduce the page link score function. We use $Link(e_i, e_j)$ to denote the number of links between entities e_i and e_j . The link score $Score_{link}(candidate_k)$ could be calculated as:

$$Score_{link}(candidate_k) = \max(Link(e_i, e_j)) \quad (6)$$

where

$$e_i \in E_{stem}, e_j \in E_{candidate_k}.$$

We only count the number of links between BAIKE documents, and it is denoted as $Score_{BDL}$

Function	Description
$Score_{BDI}$	$Score_{lexical}$ using BDI
$Score_{BPI}$	$Score_{lexical}$ using BPI
$Score_{BSI}$	$Score_{lexical}$ using BSI
$Score_{BDC}$	document level $Score_{co}$
$Score_{BPC}$	paragraph level $Score_{co}$
$Score_{BSC}$	sentence level $Score_{co}$
$Score_{BDL}$	document link score function

Table 2: Summarization of score functions.

3.1.3 Training Weights

Since we have seven score functions, we need combine them together with different weights.

For a given question, we calculate the score of every candidate as follows:

$$score_{candidate_k} = \sum_{i=1}^7 (w_i * f_i(candidate_k)) \quad (7)$$

where $k \in \{1, 2, 3, 4\}$, f_i is one of the seven score functions and w_i is the corresponding weight. Then we normalize the scores of all candidates:

$$score_k = \frac{score_{candidate_k}}{\sum_{i=1}^4 (score_{candidate_i})} \quad (8)$$

We suppose that the true answer of a question is the n -th candidate, where $n \in \{1, 2, 3, 4\}$. The loss of it is

$$loss_{question} = -\log(1 - score_n) \quad (9)$$

Now we can calculate the total loss of the dataset with M questions:

$$loss = \sum_i^M (loss_{question_i}) \quad (10)$$

All operations are derivable so that we can use gradient descent algorithm to train the weights.

3.2 NN Approach

As deep neural networks are widely used in natural language processing tasks and has gained great success, it's naturally to come up with building deep neural networks to handle GKHMC task. So, we built several deep neural networks in different structures. And, we used both TIKU and BOOK to train these models, in order to teach models not only how to answer the questions but also the historical knowledge.

To handle the joint inference between background knowledge and question stems in GKHMC

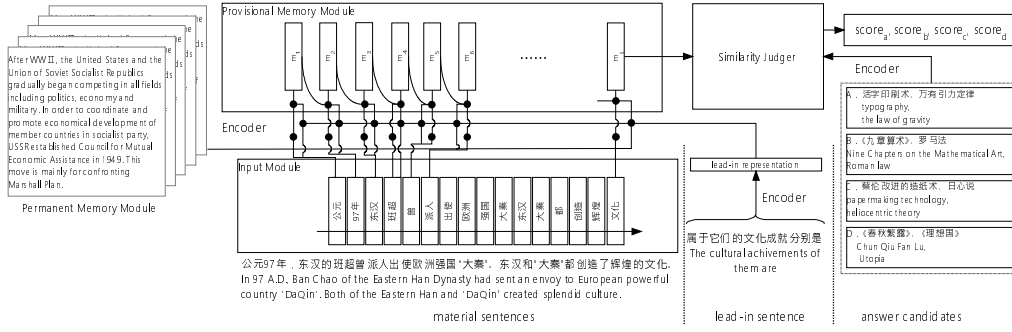


Figure 3: Diagram of PPMN. The questions hasn't been translated into English.

questions, we introduce permanent-provisional memory network (PPMN). As illuminated in Figure 3, our PPMN is composed by the following components:

1. **Permanent Memory Module** that plays the same role as a knowledge base and stores the original text from history textbooks or other relevant resource.
2. **Provisional Memory Module** that generates some contents based on the current word in background sentences, permanent knowledge and the lead-in sentence.
3. **Input Module** that reads the words sequentially in background sentences and maps them into high-dimensional vector space.
4. **Similarity Judger** that scores the similarity between the output of provisional memory and the vector representations of answer candidates.
5. **Sentence Encoder** that encodes lead-in sentence, sentences in permanent memory and answer candidates.

Permanent Memory Module: We denote the sentences encoded by sentence encoder in this module as $\{k_1, k_2, \dots, k_K\}$, where K is the scale of permanent memory. The permanent memory is a constant matrix composed by the concatenation of representation vectors of these sentences, namely $[k_1; k_2; k_3; \dots; k_K]$. Considering the time complexity of training PPMN, we only take the syllabus of all history courses including 198 sentences, i.e. $K = 198$, as the permanent memory. If necessary, all of the history text books can be taken into the permanent memory.

Provisional Memory Module: It first inquires the current word of background sentences in the permanent memory, then use an attention vector generated by current word and lead-in sentence as well as the following words to decide how to adjust itself. The update equations are as follow:

$$h_t = GRU(w_t, h_{t-1}) \quad (11)$$

$$p = softmax(pW^p h_t) \quad (12)$$

$$M_t = \sum_{i=1}^K p_i k_i \quad (13)$$

$$x = [h_t, M_t, l, h_t \circ l, M_t \circ l, h_t \circ M_t, |h_t - l|, |M_t - l|, |h_t - M_t|] \quad (14)$$

$$g = \sigma(W^g tanh(W^t x + b^t) + b^g) \quad (15)$$

$$m_t = g \circ M_t + (1 - g) \circ m_{t-1} \quad (16)$$

In the above equations, w_t denotes the t -th word in the background sentences, GRU is defined in equation (19-22), h_{t-1} and h_t are the hidden representation of w_{t-1} and w_t respectively, l stands for the lead-in sentence encoded by the sentence encoder, \circ is element-wise multiplication and m_t is the computational result of current step. The final output of this module is the last provisional memory vector m_n where n is the length of background sentences.

Input Module: This module takes the same weight matrices in sentence encoder and calculates the hidden states of every word sequentially. All the words in background sentences are first mapped into the hidden states in this module and then can be taken as input by other modules. The calculation of hidden states are the same as equation (19-22).

Similarity Judge: This module takes the concatenation of the output from provisional memory and representation of answer candidate as input and use a classifier based on logistic regression to score it. The judging procedure is defined as follow:

$$\hat{p} = \sigma(W^l[m_K; a] + b^l) \quad (17)$$

$$score = softmax(\hat{p}) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (18)$$

where W^l is a matrix that can map the concatenation vector $[m_K; a]$ into a vector \hat{p} of length 2 and a stands for the answer candidate encoded by sentence encoder.

Sentence Encoder: We experimented several recurrent neural networks with different structures as the sentence encoder. Both of Long-Short Term Memory (LSTM)(Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU)(Cho et al., 2014) perform much better than the standard *tanh* RNN. However, considering that the computation of LSTM is more complicated and time-consuming, we choose GRU as the sentence encoder. The calculation of GRU denoted as $h_t = GRU(w_t, h_{t-1})$ is as follow:

$$z = \sigma(W^z w_t + U^z w_t + b^z) \quad (19)$$

$$r = \sigma(W^r w_t + U^r w_t + b^r) \quad (20)$$

$$s = tanh(W^s w_t + U^s(r \circ h_{t-1}) + b^s) \quad (21)$$

$$h_t = (1 - z) \circ s + z \circ h_{t-1} \quad (22)$$

In the above equations, w_t is extracted from a word embedding matrix W_e initialized by word2vec(Mikolov et al., 2013) through an id number that indicates which word it is.

Loss Function: Intuitively, as we want to encourage the score as same to the true score (0 or 1) as possible, a negative log-likelihood loss function is introduced:

$$L = -\log(\hat{p}y) \quad (23)$$

where y would be $[0 \ 1]^T$ if a is the right answer or $[1 \ 0]^T$ otherwise.

Optimization Algorithm: We use the AdaDelta introduced by (Zeiler, 2012) to minimize the loss L , and use back propagation through time to optimize the calculation results of intermediate results.

	Accuracy
EQ- W_{EQ}	49.38%
SQ- W_{SQ}	28.60%

Table 3: Accuracy of SQs and EQs with their corresponding best weights.

4 Experiment

4.1 Experiments of IR Approach

To find the best weights for EQ and SQ, We use TIKU as the training dataset. Using gradient descent to optimize parameters, we get the best weights for EQs and SQs separately, that is, W_{EQ} is the weight best for EQs and W_{SQ} is the weight best for SQs. We test the weights on EQs and SQs of GKHMC with their corresponding weights, and result is shown in Table 3. As we can see, with these weights, we achieve promising result.

We use GKHMC as the dataset to test the performance of IR approach with naive bayes classifier. The precision of EQs and SQs are 48.75%, 28.42% respectively. It's clear that the accuracy of both EQs and SQs decreased with automatic classification. But still, IR approach achieves much better results on EQs than SQs.

4.2 Results of NN Approach

We take some other neural network models with memory capability as our baseline models including the standard *tanh* recurrent neural network(RNN), long-short term memory network(LSTM)(Hochreiter and Schmidhuber, 1997), gated recurrent unite(GRU)(Cho et al., 2014), end-to-end memory network(MemNN)(Sukhbaatar et al., 2015) and dynamic memory network(DMN)(Kumar et al., 2016). As for our PPMN, we summarize the syllabus of all history textbooks for senior school students to cover as much knowledge points as possible and we get 198 sentences which are taken into the permanent memory module. For all the above models, we used rmsprop(Hinton et al., 2012) with 0.001 as the learning rate to train them, the size of hidden units as well as the size of memory were both set to 400 and the size of batches were set to 1000. Also, we used dropout(Srivastava et al., 2014) to prevent the models from overfitting and the probability of it was set to 0.5. We test all these models and the results are shown in Table 4.

From the result, we observe that our PPMN

Model	EQs	SQs	All
RNN	36.25%	29.74%	31.18%
LSTM	40.63%	40.41%	40.46%
GRU	40.63%	40.24%	40.32%
MemNN	43.75%	36.13%	37.77%
DMN	44.38%	45.38%	45.16%
PPMN	45.63%	45.72%	45.70%
Random	25.00%	25.00%	25.00%

Table 4: Results of all neural network models.

gains best performance on all kinds of GKHMC questions and all memory-capable neural network models beat RNN. It’s interesting that MemNN performs much worse than other memory-capable models on SQs whereas it shows promising capability on EQs.

4.3 Combine IR Approach and NN Approach

It can be easily observed from the above experiments that IR approach and NN approach are some kind of complementary, namely they performs better to each other on different categories of questions. So, we combine the two approaches together via a weights matrix $W^c \in \mathbb{R}^{2 \times 2}$ as follows:

$$score_{EQ} = W_1^c \begin{bmatrix} score_{IR} \\ score_{NN} \end{bmatrix} \quad (24)$$

$$score_{SQ} = W_2^c \begin{bmatrix} score_{IR} \\ score_{NN} \end{bmatrix} \quad (25)$$

where the W_i^c means the i -th row of W^c and $score_{IR}, score_{NN}$ are the scores calculated by IR and NN approaches respectively. Here, the categories of questions are given by the naive bayes classifier. The performance of combined model and its comparison to the two individual approaches are illustrated in Figure 4.

4.4 Discussion

From the global aspect, it can be easily observed that IR approach are more proficient on EQs(49.38% vs 40.63%), whereas NN approach expand superior to it on SQs(28.60% vs 40.24%). And the hybrid method composed by two approaches get the best performance(42.60%).

As for the IR approach itself, the performance on EQs is much better than on SQs. This may be because that IR approach is based on the relevance between candidates and question stem. In EQs, the information given by the question stem is usually the description of the key entity which only

disappeared in the right candidate. So it’s easy for the correct candidate to achieve a higher relevance score than others. And, that’s why IR approach achieves promising result on EQs. Whereas, in SQs, the key entity doesn’t appear in any candidate. And, it needs to be inferred out from question stem. No matter in aspect of lexical matching, entity co-occurrence or page link, the relevance between question stem and correct candidate may be as low as other candidates. Therefore, it’s not surprised that IR approach is not sufficient to figure out the right choice on SQs. After adding the classifier in IR approach, we notice the decrease of accuracy on both EQs and SQs. This is because of the misclassification on the questions, which demonstrates that the weights W_{EQ}, W_{SQ} are particularly efficient on EQs, SQs.

The experiment of NN approach declared that our PPMN does show its advantages on GKHMC questions. During the training, the performance of RNN model is labile, i.e. the precision are still variational when loss is convergent. In contrast, other model’s performance is more stable. Hence, we consider that the memory mechanism helps model to “remember” the knowledge that appeared in the training data. Compared with the “inside”⁴ memory of LSTM and GRU, the specially designed memory component in MemNN, DMN and PPMN are more powerful to find out the relationships between the question stem and answer candidates in GKHMC questions. However, the limited performance of MemNN on SQs indicates that the sequences of words in GKHMC questions are especially important for questions containing no distinct entities. Last but not least, the best performance of PPMN may due highly on the novel permanent memory module which can helps finding the implicit relationships with the stored background knowledge.

The state-of-the-art performance of hybrid method indicates that combination of IR approach and NN approach is the best strategy to address the GKHMC questions. As illustrated in Figure 4, the combined method shows its enormous advantage on EQs. This may because both character and word embedding are more sufficient to cover the lexical meaning. And, some of EQs may be more suitable to be handled as SQs. Compared to the NN approach separately, the hybrid way does

⁴ We consider that the memory of LSTM and GRU are kind of stored inside the weight matrices.

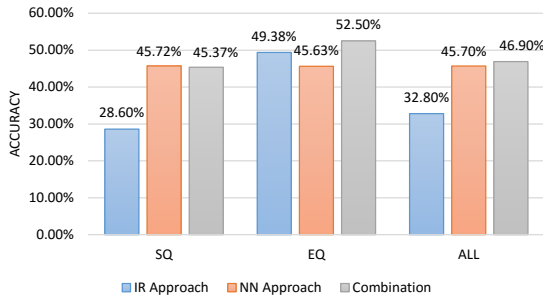


Figure 4: Result of different methods.

a little poorly on SQs, which may be caused by the loss of classification.

5 Related Work

Answering real world questions in various subjects already gained attention from the beginning of this century. The ambitious Project Halo (Friedland et al., 2004) was proposed to create a "digital" Aristotle that can encompass most of the world's scientific knowledge and be capable of addressing complex problems with novel answers. In this project, (Angele et al., 2003) employed hand-crafted rules to answer chemistry questions, (Gunning et al., 2010) took the physics and biology into account. Another important trial is solving the mathematical questions. (Mukherjee and Garain, 2008) attempted to answer them via transforming the natural language description into formal queries with hand-crafted rules, whereas recent works (Hosseini et al., 2014) started to employ learning techniques. However, none of these methods are suitable for history questions which require large background knowledge, the same to the Aristo Challenge (Clark, 2015) focused on Elementary Grade Tests which is for 6-11 year olds.

The Todai Robot Project (Fujita et al., 2014) aims to build a system that can pass the University of Tokyo's entrance examination. As part of this project, (Kanayama et al., 2012) mainly focus on addressing the yes-no questions via determining the correctness of the original proposition, and (Miyao et al., 2012) mainly focus on recognizing textual entailment between a description in Wikipedia and each option of question. But, these two methods are separated for different kinds of questions and none of them introduced neural network approach.

It's inevitable to compare the GKHMC with the factoid questions. (Berant and Liang, 2014) takes

the question as a kind of semantic parsing which can not handle the specific expressions with lots of background knowledge. Although (Yih et al., 2015) employed knowledge base, but still failed on multiple sentences questions which is beyond the scope of semantic parsing. However, the diversity of candidates in GKHMC makes these models fail to match the question with the right candidate. Another nonnegligible task is machine comprehension, also called reading comprehension. Although in several different datasets introduced by (Smith et al., 2008; Richardson et al., 2013; Weston et al., 2015), questions are open-domain and candidates may be entities or sentences, understanding these questions don't require as much background knowledge as in GKHMC and these models cannot handle the joint inference between the background knowledge and words in questions.

We are not the first to take up the Gaokao challenge, but former information retrieval approach doesn't fit to part of the questions in GKHMC and resources in their system are limited. In contrast, we introduced two different approaches to this task, compared their performance on different types of questions, combined them and gained a state-of-the-art result.

6 Conclusion and Future Work

In this work, we detailed the multiple choice questions in subject History of Gaokao, present two different approaches to address them and compared these approaches' performance on all categories of questions. We find that the IR approach are more sufficient on EQs cause the words in these questions are usually the description of right answer, whereas the NN approach performs much better on SQs, and this may be because neural network models can find out the semantic relationship between questions and candidates. When combining them together, we get the state-of-the-art performance on GKHMC, better than any individual approach. This points out that combining different approaches may be a better method to deal with the real-world questions.

In future work, we will explore whether key-value memory network proposed by (Miller et al., 2016) can help improve the performance of PPMN, what content in textbook or encyclopedia should be taken into the permanent memory, how to mathematically organize the permanent mem-

ory to make it can be reasoned on as well as whether transforming the knowledge described in natural language into formal representation is beneficial. As a long-term goal, it's necessary to introduce discourse analysis, semantic parsing to help the model truly understand the material sentences, questions and candidates.

Acknowledgments

We thank for the anonymous reviewers for helpful comments. This work was supported by the National High Technology Development 863 Program of China (No.2015AA015405) and the Natural Science Foundation of China (No.61533018). And this research work was also supported by Google through focused research awards program.

References

- Jürgen Angele, Eddie Mönch, Henrik Oppermann, Steffen Staab, and Dirk Wenke. 2003. Ontology-based query and answering in chemistry: Ontonova project Halo. In *The Semantic Web-ISWC 2003*, pages 913–928. Springer, Sanibel Island, Florida, USA.
- Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland, USA. Association for Computational Linguistics.
- Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.
- Rudi Cilibrasi and Paul Vitányi. 2007. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383.
- Peter Clark. 2015. Elementary school science and math tests as a driver for AI: Take the Aristo challenge! In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 4019–4021, Austin, Texas, USA.
- Noah Friedland, Paul Allen, Gavin Matthews, Michael Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jürgen Angele, Steffen Staab, et al. 2004. Project Halo: Towards a digital Aristotle. *AI magazine*, 25(4):29.
- Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. 2014. Overview of Todai robot project and evaluation framework of its nlp-based problem solving. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland. European Language Resources Association (ELRA).
- David Gunning, Vinay Chaudhri, Peter Clark, Ken Barker, Shaw-Yi Chaw, Mark Greaves, Benjamin Grosz, Alice Leung, David McDonald, Sunil Mishra, et al. 2010. Project Halo update-progress toward digital Aristotle. *AI Magazine*, 31(3):33–58.
- Geoffrey Hinton, Nirsh Srivastava, and Kevin Swersky. 2012. Lecture 6a overview of mini-batch gradient descent. *Coursera Lecture slides* <https://class.coursera.org/neuralnets-2012-001/lecture>.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Javad Mohammad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.
- Hiroshi Kanayama, Yusuke Miyao, and John Prager. 2012. Answering yes/no questions via question inversion. In *Proceedings of COLING 2012*, pages 1377–1392, Mumbai, India. The COLING 2012 Organizing Committee.
- Oleksandrs Kolomiyet and Marie-Francine Moens. 2011. A survey on question answering technology from an information retrieval perspective. *Information Science*, 181(24):5412–5434.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. pages 1378–1387, New York City, New York, USA. ACM.
- Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computing Research Repository*, abs/1301.3781.
- Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin,

- Texas, USA. Association for Computational Linguistics.
- Yusuke Miyao, Hideki Shima, Hiroshi Kanayama, and Teruko Mitamura. 2012. Evaluating textual entailment recognition for university entrance examinations. *ACM Transactions on Asian Language Information Processing (TALIP)*, 11(4):13.
- Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.
- Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: bringing order to the web. Technical Report 1, Stanford InfoLab.
- Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, and Roser Morante. 2013. Qa4mre 2011-2013: Overview of question answering for machine reading evaluation. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 303–320. Springer.
- Matthew Richardson, J.C. Christopher Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.
- Noah A. Smith, Michael Heilman, and Rebecca Hwa. 2008. Question generation as a competitive undergraduate course project. In *Proceedings of the NSF Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, Massachusetts, USA.
- Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, Montréal, Canada.
- Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *Computing Research Repository*, abs/1502.05698.
- Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.
- Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *Computing Research Repository*, abs/1212.5701.

If You Can't Beat Them Join Them: Handcrafted Features Complement Neural Nets for Non-Factoid Answer Reranking

Dasha Bogdanova, Jennifer Foster, Daria Dzendzik and Qun Liu

ADAPT Centre

School of Computing, Dublin City University

Dublin, Ireland

firstname.lastname@adaptcentre.ie

Abstract

We show that a neural approach to the task of non-factoid answer reranking can benefit from the inclusion of tried-and-tested handcrafted features. We present a novel neural network architecture based on a combination of recurrent neural networks that are used to encode questions and answers, and a multilayer perceptron. We show how this approach can be combined with additional features, in particular, the discourse features presented by Jansen et al. (2014). Our neural approach achieves state-of-the-art performance on a public dataset from Yahoo! Answers and its performance is further improved by incorporating the discourse features. Additionally, we present a new dataset of Ask Ubuntu questions where the hybrid approach also achieves good results.

1 Introduction

The task of Question Answering (QA) is arguably one of the oldest tasks in Natural Language Processing (NLP), attracting high levels of interest from both industry and academia. The QA track at the Text Retrieval Evaluation Conference (TREC) was introduced in 1999 and since then has encouraged many research studies by providing a platform for evaluation and making labeled datasets available. However, most research has focused on factoid questions, e.g. the TREC questions *What is the name of the managing director of Apricot Computer?* and *What was the monetary value of the Nobel Prize in 1989?* The TREC QA track organizers took care to “select questions with straightforward, obvious answers” (Voorhees and Tice, 1999) to facilitate manual assessment. In contrast, research on answering non-factoid (NF)

questions, such as manner, reason, difference and opinion questions, has been rather piecemeal. This was largely due to the absence of available labeled data for the task. This is changing, however, with the growing popularity of Community Question Answering (CQA) websites, such as Quora,¹ Yahoo! Answers² and the Stack Exchange³ family of forums.

One of the main components of a non-factoid question answering system is the answer reranking module. Given a question, it aims to rearrange the answers in order to boost the community-selected best answer to the top position. Most previous attempts to perform non-factoid answer reranking on CQA data are supervised, feature-based, learning-to-rank approaches (Jansen et al., 2014; Fried et al., 2015; Sharp et al., 2015). These methods represent the candidate answers as meaningful handcrafted features based on syntactic, semantic and discourse parses (Surdeanu et al., 2011; Jansen et al., 2014), web correlation (Surdeanu et al., 2011), and translation probabilities (Fried et al., 2015; Surdeanu et al., 2011). The resulting feature vectors are then passed to a supervised ranking algorithm, such as SVMrank (Joachims, 2006), which ranks the candidates.

There has been a recent shift in Natural Language Processing towards neural approaches involving minimal feature engineering. Several recent studies present purely neural approaches to answer reranking, with most of them focusing on the task of passage-level answer selection (dos Santos et al., 2016; Tan et al., 2015), rather than answer reranking in CQA websites (Bogdanova and Foster, 2016). These neural approaches aim to obviate the need for any feature engineering and instead focus on developing a neural architecture

¹<http://quora.com>

²<http://answers.yahoo.com>

³<http://stackexchange.com>

that learns the representations and the ranking. However, while it is possible to view a purely neural approach as an alternative to machine learning involving domain knowledge in the form of handcrafted features, there is no reason why the two approaches cannot be applied in tandem. In this paper we show that handcrafted features which encode information about discourse structure can be used to improve the performance of a neural approach to CQA answer reranking.

First, we present a novel neural approach to answer reranking that achieves competitive results on a public dataset of Yahoo! Answers (YA) that was previously introduced by Jansen et al. (2014) and later used in several other studies (Fried et al., 2015; Sharp et al., 2015; Bogdanova and Foster, 2016). Our approach is based on a combination of recurrent neural networks (RNN) and a multilayer perceptron (MLP) that receives the encodings produced by the RNNs and *interaction transformation features* that are based on the outputs of the RNNs and which aim to represent the semantic interaction between the encoded sequences. We also show how this approach can be combined with discourse features previously shown to be beneficial for the task of answer reranking.

The previous best result on the YA dataset – 37.17 P@1 and 56.82 MRR – is reported by Bogdanova and Foster (2016). Our approach achieves similar performance – 37.13 P@1 and 57.56 MRR. In contrast to the (Bogdanova and Foster, 2016) approach, which is also purely neural but requires a large in-domain corpus for pretraining, our model requires only a relatively small training set and no pretraining. The hybrid approach that includes the discourse features outperforms the neural approach on the same dataset and achieves 38.74 P@1 and 58.37 MRR. We also report experiments on a new dataset of Ask Ubuntu⁴ questions and answers. The model shows good performance on this dataset too, with the hybrid approach being about 2% more accurate in terms of P@1 than the neural approach on its own. Our error analysis provides insights into the main challenges posed by answer reranking in CQAs. These are the subjective nature of both the questions and the user choice of the best answer.

The main contributions of this paper are as follows: 1) we propose a novel neural approach for non-factoid answer reranking that achieves state-

of-the-art performance on a public dataset of Yahoo! Answers; 2) we combine this approach with an approach based on discourse features that was introduced by Jansen et al. (2014), with the hybrid approach outperforming the neural approach and the previous state-of-the-art; 3) we introduce a new dataset of Ask Ubuntu questions and answers.

This paper is organized as follows: an overview of previous work on non-factoid question answering is provided in Section 2, our neural architecture is introduced in Section 3, the discourse features that are incorporated into our neural approach are described in Section 4, the results of our experiments with these new models are presented and analysed in Section 5, and suggestions for further research are provided in Section 6.

2 Related Work

Previous work on supervised non-factoid answer reranking on CQA datasets focused mainly on feature-rich approaches. Surdeanu et al. (2011) show that CQAs such as Yahoo! Answers are a good source of knowledge for non-factoid QA. They employ four types of features in their answer reranking model: (1) similarity features: the similarity between a question and an answer based on the length-normalized BM25 formula (Robertson et al., 1994); (2) translation features: probability of the question being a translation of the answer computed using IBM’s Model 1 (Brown et al., 1993); (3) features measuring frequency and density of the question terms in the answer, such as the number of non-stop question words in the answer, the number of non-stop nouns, verbs and adjectives in the answer that do not appear in the question and tree kernel values for question and answer syntactic structures; (4) web correlation features based on Corrected Conditional Probability (Magnini et al., 2002) between the question and the answer. They explore these features both separately and in combination and find that the combination of all four feature types is most beneficial for answer reranking models.

Jansen et al. (2014) describe answer reranking experiments on YA using a diverse range of lexical, syntactic and discourse features. In particular, they show how discourse information can complement distributed lexical semantic information obtained with a skip-gram model (Mikolov et al., 2013). In this paper we use their features (discussed in detail in Section 4) in combination with

⁴<http://askubuntu.com>

a neural approach. Fried et al. (2015) improve on the lexical semantic models of Jansen et al. (2014) by exploiting indirect associations between words using higher-order models.

Methods based purely on neural models have gained popularity in various areas of NLP in recent years. The main advantage of these models is that they are often able to achieve state-of-the-art results while obviating the need for manual feature engineering. These approaches have been successful in the area of question answering. Several studies proposed models based on convolution neural networks (Severyn and Moschitti, 2015; Tymoshenko et al., 2016; Feng et al., 2015) for answer sentence selection for factoid question answering and models based on combinations of convolutional and recurrent neural networks for the task of passage-level non-factoid answer reranking (Tan et al., 2015; dos Santos et al., 2016). Recurrent neural networks and memory networks were successfully applied to the task of reading comprehension (Xiong et al., 2016; Sukhbaatar et al., 2015; Weston et al., 2015). A simple purely neural approach to non-factoid answer reranking in CQAs was proposed by Bogdanova and Foster (2016). The question-answer pairs are represented with Paragraph Vector (Le and Mikolov, 2014) distributed representations, and a multilayer perceptron is used to estimate the probability of the answer being good for the given question. The approach achieves state-of-the-art results. However, it requires unsupervised pretraining of the Paragraph Vector model on a relatively big in-domain dataset.

Recently, the Wide and Deep learning model for recommendation systems was proposed (Cheng et al., 2016). This model trains a *wide* linear model based on sparse features alongside a deep neural model, thus combining the benefits of memorization provided by the former part and the generalization provided by the latter.

In this paper, we propose a hybrid approach to answer reranking. Similarly to the wide and deep model, it combines traditional feature-based and deep neural approaches. However, in this paper we enhance the neural model with discourse chunk features that were previously found useful for this task. The features are combined with a neural model that consists of two bidirectional RNNs that encode the question and the answer and a multilayer perceptron that receives the neural encodings

and the discourse features and makes the final prediction.

3 Learning to rank answers with RNNs and MLP

We illustrate our approach to answer reranking in Figure 1. Following previous research on neural answer reranking (Severyn and Moschitti, 2015; Bogdanova and Foster, 2016), we employ the pointwise approach to ranking, i.e. we cast the ranking task as a classification task. Given a question q and an answer a , we first use two separate bidirectional RNNs⁵ to encode the question and the answer. Let $(w_1^q, w_2^q, \dots, w_k^q)$ be the sequence of question words and $(w_1^a, w_2^a, \dots, w_p^a)$ be the sequence of answer words.⁶ The first RNN encodes the sequence of question words into the sequence of context vectors $(h_1^q, h_2^q, \dots, h_k^q)$, i.e.

$$f_{RNN}^q(w_i^q, \theta_q) = h_i^q \quad (1)$$

where θ_q denote the trainable parameters of the network. More specifically, the bidirectional RNN consists of two RNNs: the forward RNN that reads the question starting from the first word until the last word and encodes it as a sequence of forward context vectors $(\overrightarrow{h}_1^q, \overrightarrow{h}_2^q, \dots, \overrightarrow{h}_k^q)$, and the reverse RNN that encodes the question starting from the last word until the first word: $(\overleftarrow{h}_k^q, \overleftarrow{h}_{k-1}^q, \dots, \overleftarrow{h}_1^q)$. The resulting context vectors are concatenations of the forward and reverse context vectors at each step, i.e. $h_i^q = [\overrightarrow{h}_i^q, \overleftarrow{h}_i^q]$. As the encoded vector representation of the question, we use the concatenation of the context vectors, i.e.

$$enc^q = [h_1^q, \dots, h_k^q] \quad (2)$$

The second bidirectional RNN encodes the answer in the same way:

$$f_{RNN}^a(w_i^a, \theta_a) = h_i^a \quad (3)$$

$$enc^a = [h_1^a, \dots, h_p^a] \quad (4)$$

where θ_a denote the trainable parameters of the network. We also want to optionally explicitly encode the interaction between the question’s context vectors and the answer’s context vectors. To

⁵We use an RNN with Gated Recurrent Units (GRU) (Bahdanau et al., 2015). Using an LSTM instead provides similar results.

⁶The questions and answers have to be padded to k and p words respectively.

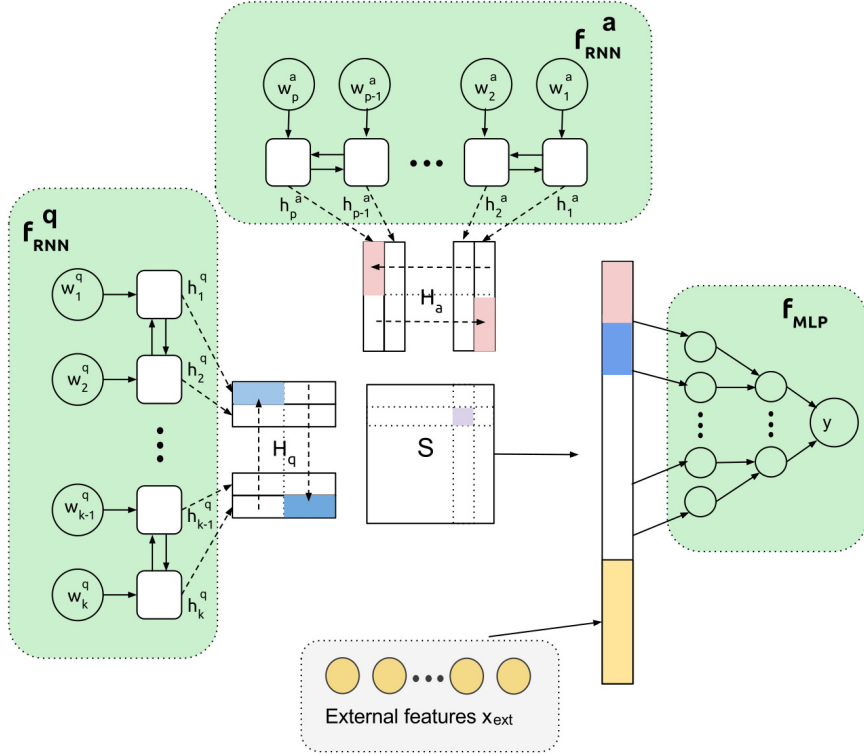


Figure 1: Our model takes a question-answer pair as an input and encodes them using separate RNNs denoted as f_{RNN}^q and f_{RNN}^a . Then a similarity matrix S over the encodings is computed and optionally concatenated with external features x_{ext} , the result is passed to a multilayer perceptron f_{MLP} that outputs the final prediction.

do this we apply the *interaction transformation* to the context vectors. More specifically, let H_q denote the matrix composed of the outputs of the question encoder RNN:

$$H_q = \begin{pmatrix} h_{1,1}^q & h_{1,2}^q & \cdots & h_{1,k}^q \\ h_{2,1}^q & h_{2,2}^q & \cdots & h_{2,k}^q \\ \vdots & \vdots & \ddots & \vdots \\ h_{d,1}^q & h_{d,2}^q & \cdots & h_{d,k}^q \end{pmatrix}$$

and H_a denote the matrix composed of the outputs of the answer RNN:

$$H_a = \begin{pmatrix} h_{1,1}^a & h_{1,2}^a & \cdots & h_{1,p}^a \\ h_{2,1}^a & h_{2,2}^a & \cdots & h_{2,p}^a \\ \vdots & \vdots & \ddots & \vdots \\ h_{d,1}^a & h_{d,2}^a & \cdots & h_{d,p}^a \end{pmatrix}$$

d is a dimensionality parameter to be experimentally tuned. We calculate the similarity matrix S between H_q and H_a , so that each element s_{ij} of the S matrix is a dot product between the corresponding encodings:

$$s_{ij} = h_i^q \cdot h_j^a$$

The similarity matrix S is unrolled and passed to the multilayer perceptron along with the question and answer encodings. They are optionally concatenated with external features x_{ext} :

$$y = f_{MLP}([S, enc^q, enc^a, x_{ext}], \theta_s) \quad (5)$$

where θ_s denote the trainable parameters of the network. The network is trained by minimizing cross-entropy:

$$L(y, \theta) = -\bar{y} \log(y) - (1 - \bar{y}) \log(1 - y)$$

where θ are all network's parameters, i.e. $\theta_q, \theta_a, \theta_s$ and \bar{y} is the true label:

$$\bar{y} = \begin{cases} 1 & \text{if } a \text{ is the best answer of the question } q \\ 0 & \text{otherwise} \end{cases}$$

4 Discourse Features

Based on the intuition that modelling question-answer structure both within and across sentences could be useful, Jansen et al. (2014) propose an answer reranking model based on discourse features

Q: How did Darth *Vader* eat?

A: *Vader* doesn't enjoy *eating* **but** he forces himself. He could *eat* with his mouth **only** inside a hyperbaric chamber.

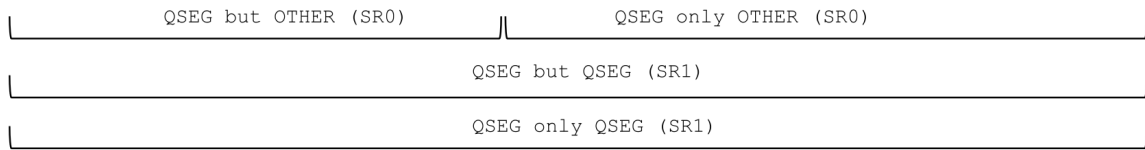


Figure 2: Feature generation for the discourse marker model of Jansen et al. (2014): first, the answer is searched for the discourse markers (in **bold**). For each discourse marker, there are several features that represent if there is an overlap (QSEG) with the question before and after the discourse marker. The features are extracted for sentence range from 0 (the same range) to 2 (two sentences before and after).

combined with lexical semantics. We experimentally evaluate these discourse features – added to our model described in Section 3 (the additional features x_{ext}) and on their own. We reuse their discourse marker model (DMM) combined with their lexical semantics model (LS). The DMM model is based on the findings of Marcu (1998), who showed that certain cue phrases indicate boundaries between elementary textual units with sufficient accuracy. These cue phrases are further referred to as discourse markers. For English, these markers include *by*, *as*, *because*, *but*, *and*, *for* and *of* – the full list can be found in Appendix B in (Marcu, 1998).

We illustrate the feature extraction process of Jansen et al. (2014) in Figure 2. First, the answer is searched for discourse markers. Each marker divides the text into two arguments: preceding and following the marker. Both arguments are searched for words overlapping with the question. Each feature denotes the discourse marker and whether there is an overlap with the question (QSEG) or not (OTHER) in the two arguments defined by the marker. The sentence range (SR) denotes the length (in sentences) of the marker’s arguments. For example, QSEG by OTHER SR0 means that in the sentence containing the *by* marker there is an overlap with the question before the marker and there is no overlap with the question after the marker. This results in 1384 different features. To assign values to each feature, the similarity between the question and each of the two arguments is computed, and the average similarity is assigned as the value of the feature. Jansen et al. (2014) use cosine similarity over *tf.idf* and over the vector space built with a skip-gram model (Mikolov et al., 2013). Further details

can be found in (Jansen et al., 2014).

5 Experiments

5.1 Data

In our experiments, we use two datasets from different CQAs. For comparability, we use the dataset created by Jansen et al. (2014) which contains 10K *how* questions from Yahoo! Answers. 50% of it is used for training, 25% for development and 25% for testing. Each question in this dataset contains at least four user-generated answers. Some examples can be found in Table 1. Further details about this dataset can be found in (Jansen et al., 2014).

To evaluate our approach on a more technical domain, we create a dataset of Ask Ubuntu (AU) questions containing 13K questions, of which 10K are used for training, 0.5K for development and 2.5K for testing. The Ask Ubuntu community is a part of the Stack Exchange family of forums. Forums of this family share the same interface and guidelines. They allow users to post questions and answers and to vote them up and down, resulting in every question and every answer having a score representing the votes it received. The author of the question may select the *best answer* to their question. We create the AU dataset in the same way as the YA dataset was created: for each question, we only rank answers provided in response to this question, and the answer labelled as the best by the question’s author is considered to be the correct answer. We make sure that the dataset contains only questions that have at least three user-provided answers and have the best answer selected, and that this answer has a non-negative score. Example questions from this dataset can be

Question: how do you cut onions without crying?

Gold: Use a sharp knife because if the onions are cut cleanly instead of slightly torn (because of a dull knife) they will release less of the chemical that makes you cry. Lighting a candle also helps with this, (...) I hope this helps.

Other Answers:

- Watch a comedy.
- Put onion in the chop blender
- close ur eyes...
- Sprinkle the surrounding area with lemon juice.
- Choose one of the followings after cutting the head and tail of the onion, split in half and peel off the skin. 1. Keep on chopping with your knife 2. Cut in quarters and put in choppers.

Table 1: Example question from the Yahoo! Answers dataset.

Question: Can't shutdown through terminal. When ever i use the following `sudo shutdown now; sudo reboot; sudo shutdown -h my laptop` goes on halt (...) is there something wrong with my installation?

Gold: Try the following code `sudo shutdown -P now (...) -P` Requests that the system be powered off after it has been brought down. `-c` Cancels a running shutdown. `-k` Only send out the warning messages and disable logins, do not actually bring the system down.

Other Answers:

- Try `sudo shutdown -h now` command to shutdown quickly.
- Try `init 0` init process shutdown all of the spawned processes/daemons as written in the init files

Table 2: Example question from the Ask Ubuntu dataset.

found in Table 2.

There are significant differences between the two datasets. While the Yahoo! Answers dataset has very short questions (10.8 on average) and relatively long answers (50.5 words), Ask Ubuntu questions can be very long, as they describe non-trivial problems rather than just ask questions. The average length of the Ask Ubuntu questions is 112.14 words, with the average answer being about 95 words long.

5.2 Experimental Setup

Following Jansen et al. (2014) and Fried et al. (2015), we implement two baselines: the baseline that selects an answer randomly and the candidate retrieval (CR) baseline. The CR baseline uses the same scoring as in Jansen et al. (2014): the questions and the candidate answers are represented using *tf-idf* over lemmas; the candidate answers are ranked according to their cosine similarity to the respective question. Additionally, we evaluate the discourse features described in Section 4 alone: we use them as the representation of the question-answer pairs that are then used as the input to a multilayer perceptron with five hidden layers. On the YA dataset, we also compare our results to the ones reported by Jansen et al. (2014) and by Bogdanova and Foster (2016).

The model described in Section 3 is regularized with L2-regularization and dropout. The development sets are used solely for early stopping

and hyperparameter selection. We tune the hyperparameters (learning rate, L2 regularization rate, dropout probabilities, dimensionality of the embeddings, the network architecture (the number of hidden layers and units, the use of GRU versus LSTM)) on the development sets. All neural networks use the rectified linear activation function (ReLU). The word embeddings are initialized randomly, no pretrained embeddings are used. We use the software provided by Jansen et al. (2014)⁷ to extract the discourse features described in Section 4 and referred to as x_{ext} in Section 3. These discourse features require that word embeddings be trained in order to calculate the similarity. Following Jansen et al. (2014), we train them using the skip-gram model (Mikolov et al., 2013) We use the L6 Yahoo dataset⁸ to train the skip-gram model for the YA dataset and the Ask Ubuntu September 2015 data dump for the AU dataset. The neural model described in Section 3 does not require pre-training of word embeddings, the embeddings are used only to extract external discourse features. To evaluate all the models, we use standard implementations of P@1 and mean reciprocal rank (MRR).

5.3 Results

We experimentally evaluate the following models:

⁷<http://nlp.sista.arizona.edu/releases/acl2014/>

⁸<http://webscope.sandbox.yahoo.com/>

- **MLP-discourse:** The discourse features are extracted as described in Section 4, an MLP is used to produce the ranking;
- **GRU-MLP:** The system described in Section 3 without the interaction matrix S and any other external features (x_{ext} in Section 3 and in Figure 1);
- **GRU-MLP-Sim:** The system described in Section 3 with the interaction matrix S and no external features;
- **GRU-MLP-Sim-Discourse:** The system described in Section 3 with the interaction matrix S and the discourse features as the external features x_{ext} ;

Table 3 reports the answer reranking P@1 and MRR of the described models along with the results of the baseline systems. The models were frozen on their best development epoch, the test set had been used neither for model selection nor for parameter tuning.⁹

Table 3 shows that the discourse features on their own with an MLP (MLP-Discourse) outperform the random and the CR baselines for both datasets. They also perform better than the approach of Jansen et al. (2014) who used SVMrank with a linear kernel. This might be due to the ability of the MLP to model non-linear dependencies. Nonetheless, the MLP-Discourse approach performs worse than the approach of Bogdanova and Foster (2016), which is based on distributed representations of documents, which probably capture more information relevant to the task.

The system described in Section 3 with no interaction transformation (only the encodings are passed to the MLP) and without any external features (x_{ext} in Section 3 and in Figure 1), referred to as GRU-MLP, outperforms the CR and the Random baselines and the systems based on the discourse features. However, it performs slightly worse than the approach of (Bogdanova and Foster, 2016). One possible reason is that the latter uses a large corpus for unsupervised pretraining.

⁹We report the results obtained with a bidirectional RNN with GRU cell, MLP with 5 hidden layers (with 5120, 2048, 1024, 512, 128 units), batch size 100, learning rate 0.01, weight decay 0.0005, dropout keep probability 0.6, and the word embedding dimensionalities and RNN outputs set to 100. The questions and answers are padded: the lengths are set to 15 words for the question and 100 words for the answer in the YA dataset and 200 and 150 words for the AU dataset.

Yahoo! Answers		
Model	P@1	MRR
Random Baseline	15.74	37.40
CR Baseline	22.63	47.17
Jansen et al. (2014)	30.49	51.89
Bogdanova and Foster (2016)	37.17	56.82
MLP-Discourse	32.72*	53.54*
GRU-MLP	36.12*	56.63*
GRU-MLP-Sim	37.13*	57.56*
GRU-MLP-Sim-Discourse	38.74*	58.37*
Ask Ubuntu		
Model	P@1	MRR
Random Baseline	26.60	53.64
CR Baseline	35.36	60.17
MLP-Discourse	37.80*	61.75*
GRU-MLP	38.56*	62.53*
GRU-MLP-Sim	39.28*	62.64*
GRU-MLP-Sim-Discourse	41.40*	64.42*

Table 3: The systems results versus the baselines. * The improvements over the CR and Random baselines are statistically significant with $p < 0.05$. All significance tests are performed with one-tailed bootstrap resampling with 10,000 iterations.

The GRU-MLP systems does not use any external data, and learns only from the small training set.

The system enriched with the interaction matrix, GRU-MLP-Sim, clearly outperforms all the baselines on both datasets, including the MLP-Discourse system. On the YA dataset, the results are better than Jansen et al. (2014) and very similar to Bogdanova and Foster (2016). On the AU dataset the improvement over the CR and the MLP-discourse systems is less remarkable, yet statistically significant. This indicates the benefit of explicitly providing the interaction features to the MLP.

The same approach with the additional discourse features described in Section 4, referred to as GRU-MLP-Sim-Discourse in Table 3, achieves the highest P@1 and MRR on the YA dataset and the AU dataset. Surprisingly, the discourse features are very helpful on the AU dataset which is highly technical, with significant parts of the information represented as commands and code.

Even though the results achieved on both datasets are similar in absolute values, the datasets are very different and the errors might be of a different nature. We provide some insights into the

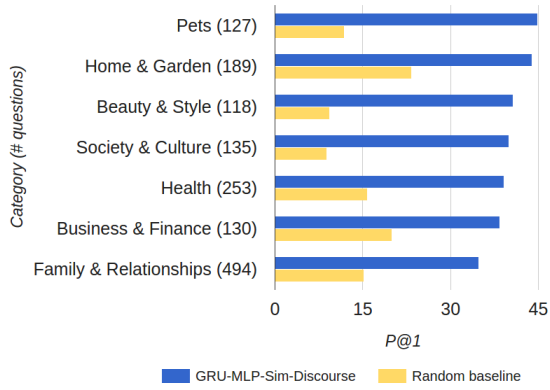


Figure 3: Average P@1 of the GRU-MLP-Sim-Discourse versus the Random baseline on the test questions from most common YA categories.

challenges raised by the two datasets in the next section.

5.4 Error Analysis

By conducting an error analysis on the YA dataset we were able to pinpoint the main causes of error as follows:

1. Despite containing only *how* questions, the dataset contains a large amount of questions asking for an **opinion** or **advice**, e.g. *How should I do my eyes?*, *How do I look?* or *How do you tell your friend you're in love with him?* rather than **information**, e.g. *How do you make homemade lasagna?* and *how do you convert avi to mpg?* About half of the questions where the best system was still performing incorrectly were of the opinion-seeking nature. This is a problem for automatic answer reranking, since the nature of the question makes it very hard to predict the quality of the answers.
2. The choice of the best answer purely relies on the user. Inspection of the data reveals that these user-provided gold labels are not always reliable. In many cases the users tend to select as the best those answers that are most **sympathetic** (see (Q1) in Table 4) or **funny** (see (Q2) and (Q3) in Table 4), rather than the ones providing more useful information.

In order to gain more insights into the reasons behind errors on the YA data, we calculated av-

erage P@1 per category.¹⁰ Figure 3 shows average P@1 of the GRU-MLP-Sim-Discourse system versus the Random baseline for the most common categories. From this figure it is clear that the most challenging category for answer reranking is Family & Relationships. This category is also the most frequent in the dataset, with 494 out of 2500 questions belonging to it. Our system achieves about 4% lower P@1 on the questions from Family & Relationships category than on the whole test set, while the random baseline performs as well as on the whole test set (the average number of answers per question in this category does not differ much from the dataset average). The low P@1 on this category is related to the reasons pointed out above: most questions in this category are of an opinion-seeking nature: *How do I know if my boyfriend really loves me?*, *How do I fix my relationship?*, *How do I find someone that loves me?*, making it hard to assess the quality of the answers.

The Ask Ubuntu dataset is rather different. In contrast to the YA dataset, which contains many subjective questions, most Ask Ubuntu questions relate to a complex technology and usually require deep domain knowledge to be answered. Moreover, many questions and answers contain code, screenshots and links to external resources. Reliably reranking such answers based on textual information alone might be an unattainable goal. The technical complexity of the questions can give rise to ambiguity. For instance, in (Q2) in Table 5 it is not clear if the question refers to the metapackage *ubuntu-desktop* or to ubuntu default packages in general. Another potential source of difficulty comes from the fact that the technologies being discussed on Ask Ubuntu change rapidly: some answers selected as best might be outdated (see (Q1) in Table 5).

6 Conclusions and Future Work

In this paper we presented a neural approach to open-domain non-factoid answer reranking. Previous studies in this area have either been feature-based or purely neural approaches that require no manual feature engineering. We show that these two approaches can be successfully combined. We propose a novel neural architecture whereby the question-answer pairs are first encoded using two

¹⁰We first mapped the low-level categories provided in the dataset to the 26 high-level YA categories. We only consider categories that contained at least 100 questions.

(Q1) How does someone impress a person during a conversation that u are as good as an oxford/harvard grad.?
(Gold) i think you're chasing down the wrong path. but hell, what do i know?
(Prediction) There are two parts. Understanding your area well, and being creative. The understanding allows you the material for your own opinions to have heft and for you to analyse the opinions of others. After that, it's just good vocabulary which comes from reading a great deal and speaking with others. Like many other endeavors practice is what makes your performance improve.
(Q2) How to get my mom to stop smoking?
(Gold) Throw a glass of water on her every time she sparks one up
(Prediction) Never nag her. Instead politely insist on your right to stay free of all the risks associated with another person's smoking. For example, do not allow her to smoke inside the car, the house or anywhere near you (...)
(Q3) How do i hip hop dance??!?!?
(Gold) Basically, you shake what your mother gave you.
(Prediction) Listen to previous freestyle flows and battles by great artists (...) Understand the techniques those artists use to flow and battle (...)

Table 4: Example incorrect predictions of the system on the Yahoo! Answers dataset.

(Q1) How do I add the kernel PPA? I can get Ubuntu mainline kernels from this kernel PPA - is there a way to add it to my repository list the same as regular Launchpad PPAs?
(Gold) Warning : This answer is outdated. As of writing this warning (6.10.2013) the kernel-ppa used here is no longer updated. Please disregard this answer. <code>sudo apt-add-repository ppa:kernel-ppa/ppa</code> <code>sudo apt-get update</code> <code>sudo apt-get install PACKAGENAME</code>
(Prediction) Since the kernel ppa is not really maintained anymore, here's a semi-automatic script: https://github.com/medigeek/kmp-downloader
(Q2) Which language is ubuntu-desktop mostly coded in? I heard it is Python
(Gold) Poked around in Launchpad: ubuntu-desktop to and browsed the source for a few mins. It appears to be a mix of Python and shell scripts.
(Prediction) I think the question referred to the language used to write the applications running on the default installation. It's hard to say which language is used the most, but i would guess C or C++. This is just a guess and since all languages are pretty equal in terms of outcome, it doesn't really matter.

Table 5: Example incorrect predictions of the system on the Ask Ubuntu dataset.

recurrent neural networks, then the interaction matrix is calculated, concatenated with external features, and passed as an input to a multilayer perceptron. As external features, we evaluate the discourse features that were found useful for this task by Jansen et al. (2014). The combined approach achieves new state-of-the-art results on two CQA datasets.

However, despite these encouraging results, the P@1 is still below 40%. As the error analysis shows, this is due to the nature of the dataset: the user choice of the best answer is not always reliable and the questions are often seeking opinions rather than information. The ceiling for this task could be very low. Manual annotation of CQA data might help in determining the upper bound.

Future work should aim to create more reliable gold standards for this task. As we show in this paper, the CQAs contain various types of question: some of which are seeking information and some not. Existing corpora of opinion questions, such as

the OpQA corpus (Stoyanov et al., 2005), could be used in future research to distinguish those from the information-seeking questions. Another possible direction for future work is in combining the neural approach with other external features, such as features based on web correlation between the question and the answer, and similarities between their syntactic structures.

Acknowledgements

This research is supported by Science Foundation Ireland through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre for Digital Content Technology. The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *5th International Conference on Learning Representations 2015*.
- Dasha Bogdanova and Jennifer Foster. 2016. This is how we do it: Answer reranking for open-domain how questions with paragraph vectors and minimal feature engineering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1290–1295, San Diego, California. Association for Computational Linguistics.
- Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.
- Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. *arXiv:1606.07792*.
- Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv:1602.03609*.
- Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE.
- Daniel Fried, Peter Jansen, Gustave Hahn-Powell, Mihai Surdeanu, and Peter Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210.
- Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 977–986, Baltimore, Maryland, June. Association for Computational Linguistics.
- Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.
- Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.
- Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. 2002. Is it the right answer?: exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 425–432. Association for Computational Linguistics.
- Daniel C. Marcu. 1998. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*. Ph.D. thesis, Toronto, Ont., Canada, Canada.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Stephen E Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *Proceedings of the 3rd Text REtrieval Conference*, pages 109–126.
- Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382.
- Rebecca Sharp, Peter Jansen, Mihai Surdeanu, and Peter Clark. 2015. Spinning straw into gold: Using free text to train monolingual alignment models for non-factoid question answering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 231–237, Denver, Colorado, May–June. Association for Computational Linguistics.
- Veselin Stoyanov, Claire Cardie, and Janyce Wiebe. 2005. Multi-perspective question answering using the opqa corpus. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 923–930. Association for Computational Linguistics.
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.
- Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Comput. Linguist.*, 37(2):351–383, June.
- Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv:1511.04108*.
- Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 1268–1278, San Diego, California, June. Association for Computational Linguistics.

Ellen M. Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. *arXiv:1603.01417*.

Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks

Rajarshi Das, Arvind Neelakantan, David Belanger, Andrew McCallum

College of Information and Computer Sciences

University of Massachusetts, Amherst

{rajarshi, arvind, belanger, mccallum}@cs.umass.edu

Abstract

Our goal is to combine the rich multi-step inference of symbolic logical reasoning with the generalization capabilities of neural networks. We are particularly interested in complex reasoning about entities and relations in text and large-scale knowledge bases (KBs). Neelakantan et al. (2015) use RNNs to obtain dense representations of multi-hop paths in KBs; however for multiple reasons, the approach lacks accuracy and practicality. This paper proposes three significant modeling advances: (1) we learn to jointly reason about relations, *entities*, and *entity-types*; (2) we use neural attention modeling to incorporate *multiple paths*; (3) we learn to *share strength in a single RNN* that represents logical composition across all relations. On a large-scale Freebase+ClueWeb prediction task, we achieve 25% error reduction, and a 53% error reduction on sparse relations. On chains of reasoning in WordNet we reduce error in mean quantile by 84% versus the previous state of the art.¹

1 Introduction

There is a rising interest in extending neural networks to perform more complex reasoning, formerly addressed only by symbolic and logical reasoning systems. So far this work has mostly focused on small or synthetic data (Grefenstette, 2013; Bowman et al., 2014; Rocktäschel and Riedel, 2016). Our interest is primarily in reasoning about large knowledge bases (KBs) with diverse semantics, populated from text. One method

¹The code and data are available at <https://rajarshd.github.io/ChainsOfReasoning/>

i.	$\text{place.birth}(a, b) \leftarrow \textit{‘was_born_in’}(a, x) \wedge \textit{‘commonly_known_as’}(x, b)$
ii.	$\text{location.contains}(a, b) \leftarrow \text{nationality}^{-1}(a, x) \wedge \text{place.birth}(x, b)$
iii.	$\text{book.characters}(a, b) \leftarrow \textit{‘aka’}(a, x) \wedge (\text{theater.character.plays})^{-1}(x, b)$
iv.	$\text{cause.death}(a, b) \leftarrow \textit{‘contracted’}(a, b)$

Table 1: Several highly probable clauses learnt by our model. The textual relations are shown in quotes and italicized. Our model has the ability to combine textual and schema relations. r^{-1} is the inverse relation r , i.e. $r(a, b) \Leftrightarrow r^{-1}(b, a)$.

for populating a KB from text (and for representing diverse semantics in the KB) is *Universal Schema* (Riedel et al., 2013; Verga et al., 2016), which learns vector embeddings of relation types - the union of all input relation types, both from the schemas of multiple structured KBs, as well as expressions of relations in natural language text.

An important reason to populate a KB is to support not only look-up-style question answering, but reasoning on its entities and relations in order to make inferences not directly stored in the KB. KBs are often highly incomplete (Min et al., 2013), and reasoning can fill in these missing facts. The “matrix completion” mechanism that underlies the common implementation of Universal Schema can thus be seen as a simple type of reasoning, as can other work in tensor factorization (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013). However these methods can be understood as operating on single pieces of evidence: for example, inferring that Microsoft-*located-in*-Seattle implies Microsoft-*HQ-in*-Seattle.

A highly desirable, richer style of reasoning

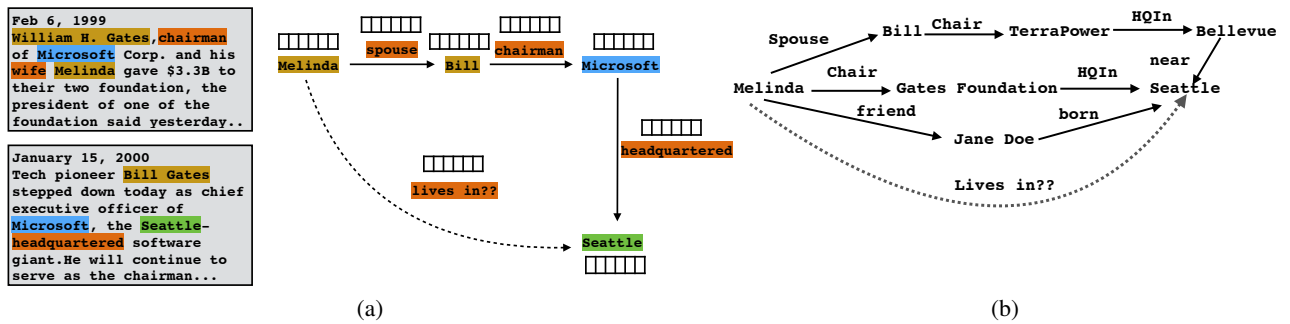


Figure 1: The nodes in the knowledge graphs represent entities and the labeled edges represent relations. (a) A path between ‘Melinda’ and ‘Seattle’ combining relations from two different documents. (b) There are multiple paths between entities in a knowledge graph. The top two paths are predictive of the fact that Melinda may ‘live in’ Seattle, but the bottom (fictitious) path isn’t.

makes inferences from Horn clauses that form multi-hop paths containing three or more entities in the KB’s entity-relation graph. For example, we may have no evidence directly linking Melinda Gates and Seattle. However, we may infer with some likelihood that Melinda–lives-in–Seattle, by observing that the KB contains the path Melinda–spouse–Bill–chairman–Microsoft–HQ-in–Seattle (Fig. 1a).

Symbolic rules of this form are learned by the Path Ranking Algorithm (PRA) (Lao et al., 2011). Dramatic improvement in generalization can be obtained by reasoning about paths, not in terms of relation-symbols, but Universal Schema style relation-vector-embeddings. This is done by Neelakantan et al. (2015), where RNNs compose the per-edge relation embeddings along an arbitrary-length path, and output a vector embedding representing the inferred relation between the two entities at the end-points of the path. This approach thus represents a key example of complex reasoning over Horn clause chains using neural networks. However, for multiple reasons detailed below it is inaccurate and impractical.

This paper presents multiple modeling advances that significantly increase the accuracy and practicality of RNN-based reasoning on Horn clause chains in large-scale KBs. (1) Previous work, including (Lao et al., 2011; Neelakantan et al., 2015; Guu et al., 2015) reason about chains of relations, but not the entities that form the nodes of the path. In our work, we jointly learn and reason about relation-types, entities, and entity-types. (2) The same previous work takes only a single path as evidence in inferring new predictions. However, as shown in Figure 1b, multiple paths can provide ev-

idence for a prediction. In our work, we use neural attention mechanisms to reason about multiple paths. We use a pooling function which does soft attention during *gradient step* and find it to work better. (3) The most problematic impracticality of the above previous work² for application to KBs with broad semantics is their requirement to train a separate model for each relation-type to be predicted. In contrast, we train a single, high-capacity RNN that can predict all relation types. In addition to efficiency advantages, our approach significantly increases accuracy because the multi-task nature of the training shares strength in the common RNN parameters.

We evaluate our new approach on a large scale dataset of Freebase entities, relations and ClueWeb text. In comparison with the previous best on this data, we achieve an error reduction of 25% in mean average precision (MAP). In an experiment specially designed to explore the benefits of sharing strength with a single RNN, we show a 54% error reduction in relations that are available only sparsely at training time. We also evaluate on a second data set, chains of reasoning in WordNet. In comparison with previous state-of-the-art (Guu et al., 2015) our model achieves a 84% reduction in error in mean quantile.

2 Background

In this section, we introduce the compositional model (Path-RNN) of Neelakantan et al. (2015). The Path-RNN model takes as input a path between two entities and infers new relations between them. Reasoning is performed non-

²with exception of (Guu et al., 2015)

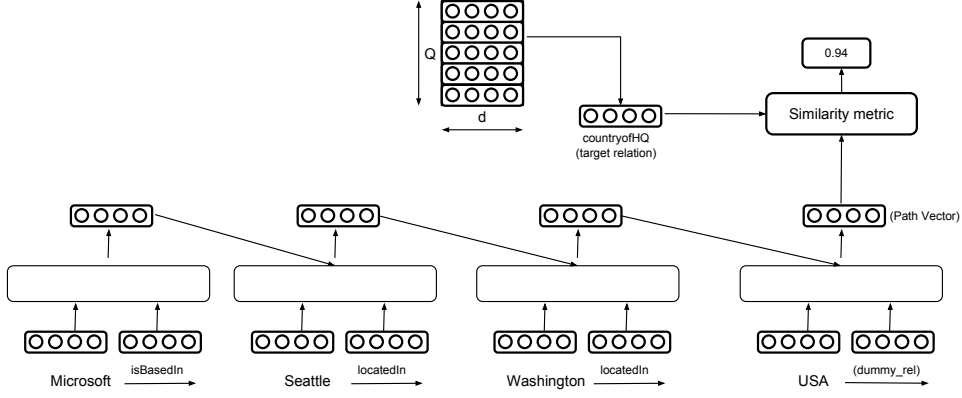


Figure 2: At each step, the RNN consumes both entity and relation vectors of the path. The entity representation can be obtained from its types. The path vector \mathbf{y}_π is the last hidden state. The parameters of the RNN and relation embeddings are shared across all query relations. The dot product between the final representation of the path and the query relation gives a confidence score, with higher scores indicating that the query relation exists between the entity pair.

atomically about conjunctions of relations in an arbitrary length path by composing them with a recurrent neural network (RNN). The representation of the path is given by the last hidden state of the RNN obtained after processing all the relations in the path.

Let (e_s, e_t) be an entity pair and \mathcal{S} denote the set of paths between them. The set \mathcal{S} is obtained by doing random walks in the knowledge graph starting from e_s till e_t . Let $\pi = \{e_s, r_1, e_1, r_2, \dots, r_k, e_t\} \in \mathcal{S}$ denote a path between (e_s, e_t) . The length of a path is the number of relations in it, hence, $(\text{len}(\pi) = k)$. Let $\mathbf{y}_{r_t} \in \mathbb{R}^d$ denote the vector representation of r_t . The Path-RNN model combines all the *relations* in π sequentially using a RNN with an intermediate representation $\mathbf{h}_t \in \mathbb{R}^h$ at step t given by

$$\mathbf{h}_t = f(\mathbf{W}_{hh}^r \mathbf{h}_{t-1} + \mathbf{W}_{ih}^r \mathbf{y}_{r_t}^r). \quad (1)$$

$\mathbf{W}_{hh}^r \in \mathbb{R}^{h \times h}$ and $\mathbf{W}_{ih}^r \in \mathbb{R}^{d \times h}$ are the parameters of the RNN. Here r denotes the query relation. Path-RNN has a specialized model for predicting each query relation r , with separate parameters $(\mathbf{y}_{r_t}^r, \mathbf{W}_{hh}^r, \mathbf{W}_{ih}^r)$ for each r . f is the sigmoid function. The vector representation of path π (\mathbf{y}_π) is the last hidden state \mathbf{h}_k . The similarity of \mathbf{y}_π with the query relation vector \mathbf{y}_r is computed as the dot product between them:

$$\text{score}(\pi, r) = \mathbf{y}_\pi \cdot \mathbf{y}_r \quad (2)$$

Pairs of entities may have several paths connecting them in the knowledge graph (Figure 1b). Let $\{s_1, s_2, \dots, s_N\}$ be the similarity scores (Equation

2) for N paths connecting an entity pair (e_s, e_t) . Path-RNN computes the probability that the entity pair (e_s, e_t) participates in the query relation (r) by,

$$\mathbb{P}(r|e_s, e_t) = \sigma(\max(s_1, s_2, \dots, s_N)) \quad (3)$$

where σ is the *sigmoid* function.

Path-RNN and other models such as the Path Ranking Algorithm (PRA) and its extensions (Lao et al., 2011; Lao et al., 2012; Gardner et al., 2013; Gardner et al., 2014) makes it impractical to be used in downstream applications, since it requires training and maintaining a model for each relation type. Moreover, parameters are not shared across multiple target relation types leading to large number of parameters to be learned from the training data.

In (3), the Path-RNN model selects the maximum scoring path between an entity pair to make a prediction, possibly ignoring evidence from other important paths. Not only is this a waste of computation (since we have to compute a forward pass for all the paths anyway), but also the relations in all other paths do not get any gradient updates during training as the max operation returns zero gradient for all other paths except the maximum scoring one. This is especially ineffective during the initial stages of the training since the maximum probable path will be random.

The Path-RNN model and other multi-hop relation extraction approaches (such as Guu et al. (2015)) ignore the entities in the path. Consider the following paths JFK-locatedIn-

NYC-locatedIn-NY and Yankee Stadium-locatedIn-NYC-locatedIn-NY. To predict the *airport_serves* relation, the Path-RNN model assigns the same scores to both the paths even though the first path should be ranked higher. This is because the model does not have information about the entities and just uses the relations in the path for prediction.

3 Modeling Approach

3.1 Shared Parameter Architecture

Previous section discussed the problems associated with per-relation modeling approaches. In response, we *share* the relation type representation and the composition matrices of the RNN across all target relations enabling lesser number of parameters for the same training data. We refer to this model as *Single-Model*. Note that this is just *multi-task learning* (Caruana, 1997) among prediction of target relation types with an underlying shared parameter architecture. The RNN hidden state in (1) is now given by:

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{ih}\mathbf{y}_{rt}). \quad (4)$$

Readers should take note that the parameters here are independent of each target relation r .

Model Training

We train the model using existing observed facts (triples) in the KB as positive examples and unobserved facts as negative examples. Let $\mathcal{R} = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$ denote the set of all query relation types that we train for. Let $\Delta_{\mathcal{R}}^+, \Delta_{\mathcal{R}}^-$ denote the set of positive and negative triples for *all* the relation types in \mathcal{R} . The parameters of the model are trained to minimize the negative log-likelihood of the data.

$$\begin{aligned} L(\Theta, \Delta_{\mathcal{R}}^+, \Delta_{\mathcal{R}}^-) = & -\frac{1}{M} \sum_{e_s, e_t, r \in \Delta_{\mathcal{R}}^+} \log \mathbb{P}(r|e_s, e_t) \\ & + \sum_{\hat{e}_s, \hat{e}_t, \hat{r} \in \Delta_{\mathcal{R}}^-} \log(1 - \mathbb{P}(\hat{r}|\hat{e}_s, \hat{e}_t)) \end{aligned} \quad (5)$$

Here M is the total number of training examples and Θ denotes the set of all parameters of the model (lookup table of embeddings (shared) and parameters of the RNN (shared)). It should be noted that the Path-RNN model has a separate loss function for each relation $r \in \mathcal{R}$ which depends only on the relevant subset of the data.

3.2 Score Pooling

In this section, we introduce new methods of score pooling that takes into account multiple paths between an entity pair. Let $\{s_1, s_2, \dots, s_N\}$ be the similarity scores (Equation 2) for N paths connecting an entity pair (e_s, e_t) . The probability for entity pair (e_s, e_t) to participate in relation r (Equation 3) is now given by,

1. Top- (k) : A straightforward extension of the ‘max’ approach in which we average the top k scoring paths. Let \mathcal{K} denote the indices of top- k scoring paths.

$$\mathbb{P}(r|e_s, e_t) = \sigma\left(\frac{1}{k} \sum_j s_j\right), \forall j \in \mathcal{K}$$

2. Average: Here, the final score is the average of scores of all the paths.

$$\mathbb{P}(r|e_s, e_t) = \sigma\left(\frac{1}{N} \sum_{i=1}^N s_i\right)$$

3. LogSumExp: In this approach the pooling layer is a smooth approximation to the ‘max’ function - LogSumExp (LSE). Given a vector of scores, the LSE is calculated as

$$\text{LSE}(s_1, s_2, \dots, s_N) = \log\left(\sum_i \exp(s_i)\right)$$

and hence the probability of the triple is,

$$\mathbb{P}(r|e_1, e_2) = \sigma(\text{LSE}(s_1, s_2, \dots, s_N))$$

The average and the LSE pooling functions apply non-zero weights to *all* the paths during inference. However only a few paths between an entity pair are predictive of a query relation. LSE has another desirable property since $\frac{\partial \text{LSE}}{\partial s_i} = \frac{\exp(s_i)}{\sum_i \exp(s_i)}$. This means that during the back-propagation step, every path will receive a share of the gradient proportional to its score and hence this is a kind of attention during the gradient step. In contrast, for averaging, every path will receive equal ($\frac{1}{N}$) share of the gradient. Top- (k) (similar to max) receives sparse gradients.

3.3 Incorporating Entities

A straightforward way of incorporating entities is to include entity representations (along with relations) as input to the RNN. Learning separate

representations of entity, however has some disadvantages. The distribution of entity occurrence is heavy tailed and hence it is hard to learn good representations of rarely occurring entities. To alleviate this problem, we use the entity types present in the KB as described below.

Most KBs have annotated types for entities and each entity can have multiple types. For example, Melinda Gates has types such as *CEO*, *Duke University Alumni*, *Philanthropist*, *American Citizen* etc. We obtain the entity representation by a simple addition of the entity type representations. The entity type representations are learned during training. We limit the number of entity types for an entity to 7 most frequently occurring types in the KB. Let $\mathbf{y}_{e_t} \in \mathbb{R}^m$ denote the representation of entity e_t , then 4 now becomes

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{ih}\mathbf{y}_{r_t} + \mathbf{W}_{eh}\mathbf{y}_{e_t}) \quad (6)$$

$\mathbf{W}_{eh} \in \mathbb{R}^{m \times h}$ is the new parameter matrix for projecting the entity representation. Figure 2 shows our model with an example path between entities (*Microsoft*, *USA*) with *country-OfHQ* (country of head-quarters) as the query relation.

4 Related Work

Two early works on extracting clauses and reasoning over paths are SHERLOCK (Schoenmackers et al., 2010) and the Path Ranking Algorithm (PRA) (Lao et al., 2011). SHERLOCK extracts purely symbolic clauses by exhaustively exploring relational paths of increasing length. PRA replaces exhaustive search by random walks. Observed paths are used as features for a per-target-relation binary classifier. Lao et al. (2012) extend PRA by augmenting KB-schema relations with observed text patterns. However, these methods do not generalize well to millions of distinct paths obtained from random exploration of the KB, since each unique path is treated as a singleton, where no commonalities between paths are modeled. In response, pre-trained vector representations have been used in PRA to tackle the feature explosion (Gardner et al., 2013; Gardner et al., 2014) but still rely on a classifier using atomic path features. Yang et al. (2015) also extract horn rules, but they restrict it to a length of 3 and the literals are restricted to schema types in the knowledge base. Zeng et al. (2016) show improvements in relation extraction by incorporating sentences which

Stats	#
# Freebase relation types	27,791
# textual relation types	23,599
# query relation types	46
# entity pairs	3.22M
# unique entity types	2218
Avg. path length	4.7
Max path length	7
Total # paths	191M

Table 2: Statistics of the dataset.

contain one entity.

Guu et al. (2015) introduce new compositional techniques by modeling additive and multiplicative interactions between relation matrices in the path. However they model only a *single* path between an entity pair in-contrast to our ability to consider multiple paths. Toutanova et al. (2016) improves upon them by additionally modeling the intermediate entities in the path and modeling multiple paths. However, in their approach they have to store scores for intermediate path length for *all* entity pairs, making it prohibitive to be used in our setting where we have more than 3M entity pairs. They also model entities as just a scalar weight whereas we learn both entity and type representations. Lastly it has been shown by Neelakantan et al. (2015) that non-linear composition function out-performs linear functions (as used by them) for relation extraction tasks.

The performance of relation extraction methods have been improved by incorporating entity types for their candidate entities, both in sentence level (Roth and Yih, 2007; Singh et al., 2013) and KB relation extraction (Chang et al., 2014), and in learning entailment rules (Berant et al., 2011). Serban et al. (2016) use RNNs to generate factoid question from Freebase.

5 Results

Data and Experimental Setup

We apply our models to the dataset released by Neelakantan et al. (2015), which is a subset of Freebase enriched with information from ClueWeb. The dataset is comprised of a set of triples (e_1, r, e_2) and also the set of paths connecting the entity pair (e_1, e_2) in the knowledge graph. The triples extracted from ClueWeb consists of sentences that contain entities linked to Freebase (Orr et al., 2013). The raw text between the two entities in the sentence forms the relation

type. To limit the number of textual relations, we retain the two following words after the first entity and two words before the second entity. We also collect the entity type information from Freebase. Table 2 summarizes some important statistics. For the PathQA experiment, we use the same train/dev/test split of WordNet dataset released by Guu et al. (2015) and hence our results are directly comparable to them. The WordNet dataset has just 22 relation types and 38194 entities which is order of magnitudes less than the dataset we use for relation extraction tasks.

The dimension of the relation type representations and the RNN hidden states are $d, h = 250$ and the entity and type embeddings have $m = 50$ dimensions. The Path-RNN model has sigmoid units as their activation function. However, we found rectifier units (ReLU) to work much better (Le et al., 2015), even when compared to LSTMs (73.2 vs 72.4 in MAP). For the path-query experiment, the dimension of entity, relation embeddings and hidden units are set to 100 (as used by Guu et al. (2015)). As our evaluation metric, we use the average precision (AP) to score the ranked list of entity pairs. The MAP score is the mean AP across all query relations. AP is a strict metric since it penalizes when an incorrect entity is ranked higher above correct entities. Also MAP approximates the area under the Precision Recall curve (Manning et al., 2008). We use Adam (Kingma and Ba, 2014) for optimization for all our experiments with the default hyperparameter settings (learning rate = $1e^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-8}$). Statistical significance for scores reported in Table 3 were done with a paired- t test.

5.1 Effect of Pooling Techniques

Section 1 of Table 3 shows the effect of the various pooling techniques presented in section 3.2. It is encouraging to see that *LogSumExp* gives the best results. This demonstrates the importance of considering information from all the paths. However, Avg. pooling performs the worst, which shows that it is also important to weigh the paths scores according to their values. Figure 3 plots the training loss w.r.t gradient update step. Due to non-zero gradient updates for all the paths, the LogSumExp pooling strategy leads to faster training vs. max pooling, which has sparse gradients. This is especially relevant during the early stages of training, where the argmax path is essentially a random

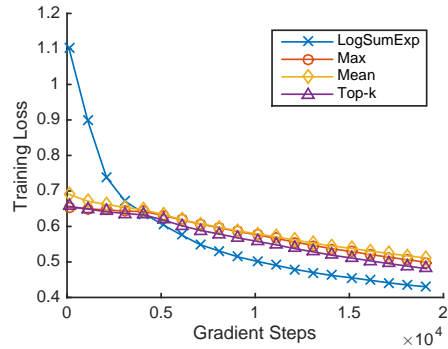


Figure 3: Comparison of the training loss w.r.t gradient update steps of various pooling methods. The loss of LogSumExp decreases the fastest among all pooling methods and hence leads to faster training.

guess. The scores of max and LSE pooling are significant with ($p < 0.02$).

5.2 Comparison with multi-hop models

We next compare the performance of the Single-Model with two other multi-hop models - Path-RNN and PRA(Lao et al., 2011). Both of these approaches train an individual model for each query relation. We also experiment with another extension of PRA that adds bigram features (PRA + Bigram). Additionally, we run an experiment replacing the max-pooling of Path-RNN with LogSumExp. The results are shown in the second section of Table 3. It is not surprising to see that the Single-Model, which leverages parameter sharing improves performance. It is also encouraging to see that LogSumExp makes the Path-RNN baseline stronger. The scores of Path-RNN (with LSE) and Single-Model are significant with ($p < 0.005$).

5.3 Effect of Incorporating Entities

Next, we provide quantitative results supporting our claim that modeling the entities along a KB path can improve reasoning performance. The last section of Table 3 lists the performance gain obtained by injecting information about entities. We achieve the best performance when we represent entities as a function of their annotated types in Freebase (Single-Model + Types) ($p < 0.005$). In comparison, learning separate representations of entities (Single-Model + Entities) gives slightly worse performance. This is primarily because we encounter many new entities during test time, for

Model	Performance (%MAP)	Pooling
Single-Model	68.77	Max
Single-Model	55.80	Avg.
Single-Model	68.20	Top(k)
Single-Model	70.11	LogSumExp
PRA	64.43	n/a
PRA + Bigram	64.93	n/a
Path-RNN	65.23	Max
Path-RNN	68.43	LogSumExp
Single-Model	70.11	LogSumExp
PRA + Types	64.18	n/a
Single-Model	70.11	LogSumExp
Single-Model + Entity	71.74	LogSumExp
Single-Model + Types	73.26	LogSumExp
Single-Model + Entity + Types	72.22	LogSumExp

Table 3: The first section shows the effectiveness of LogSumExp as the score aggregation function. The next section compares performance with existing multi-hop approaches and the last section shows the performance achieved using joint reasoning with entities and types.

which our model does not have a learned representation. However the relatively limited number of entity types helps us overcome the problem of representing unseen entities. We also extend PRA to include entity type information (PRA + Types), but this did not yield significant improvements.

5.4 Performance in Limited Data Regime

In constructing our dataset, we selected query relations with reasonable amounts of data. However, for many important applications we have very limited data. To simulate this common scenario, we create a new dataset by randomly selecting 23 out of 46 relations and removing all but 1% of the positive and negative triples previously used for training.

Effectively, the difference between Path-RNN and Single-Model is that Single-Model does multitask learning, since it shares parameters for different target relation types. Therefore, we expect it to outperform Path-RNN on this small dataset, since this multitask learning provides additional regularization. We also experiment with an extension of Single-Model where we introduce an additional task for multitask learning, where we seek to predict annotated types for entities. Here, parameters for the entity type embeddings are shared with the Single-Model. Supervision for this task is provided by the entity type annotation in the KB. We train with a Bayesian Personalized Ranking loss of Rendle et al. (2009). The results are listed in Table 4. With Single-Model there is a clear jump in performance as we expect. The additional multitask training with types gives a very incremental gain.

Model	Performance (%MAP)
Path-RNN	22.06
Single-Model	63.33
Single-Model + MTL	64.81

Table 4: Model performance when trained with a small fraction of the data.

5.5 Answering Path Queries

Guu et al. (2015) introduce a task of answering questions formulated as path traversals in a KB. Unlike binary fact prediction, to answer a path query, the model needs to find the set of correct target entities ‘ t ’ that can be reached by starting from an initial entity ‘ s ’ and then traversing the path ‘ p ’. They model additive and multiplicative interactions of relations in the path. It should be noted that the compositional Trans-E and Bilinear-diag have comparable number of parameters to our model since they also represent relations as vectors, however the Bilinear model learns a dense square *matrix* for each relation and hence has a lot more number of parameters. Hence, we compare with Trans-E and Bilinear-diag models. Bilinear-diag has also been shown to outperform Bilinear models (Yang et al., 2015).

Instead of combining relations using simple additions and multiplications, we propose to combine the intermediate hidden representations h_i obtained from a RNN (via (4)) after consuming relation r_i at each step. Let \mathbf{h} denote the sum of all intermediate representations h_i . The score of a triple (s, p, t) by our model is given by $x_s^\top \text{diag}(\mathbf{h})x_t$ where $\text{diag}(\mathbf{h})$ represents a diagonal

Horn Clause (Body)	Without Entities	With Entities	Universal
$\text{location.contains}(x, a) \wedge \text{location.contains}(a, y)$	0.9149	0.949	Y
$(\text{person.nationality})^{-1}(x, a) \wedge \text{place.birth}(a, y)$	0.7702	0.9256	N

Table 5: Body of two clauses both of which are predictive of $\text{location.contains}(x, y)$. First fact is universally true but the truth value of the second clause depends on the value of the entities in the clause. The model without entity parameters cannot discriminate this and outputs a lower overall confidence score.

Model	MQ
Comp. Bilinear Diag	90.4
Comp. Trans-E	93.3
Our Model	98.94

Table 6: Performance on path queries in WordNet.

matrix with vector \mathbf{h} as its diagonal elements.

We compare to the results reported by Guu et al. (2015) on the WordNet dataset. It should be noted that the dataset is fairly small with just 22 relation types and an average path length of 3.07. More importantly, there are only few *unseen paths* during test time and only *one* path between an entity pair, suggesting that this dataset is not an ideal test bed for compositional neural models. The results are shown in table 6. Mean Quantile(MQ) is the fraction of incorrect entities which have been scored lower than the correct entity. Our model achieves a 84% reduction in error when compared to their best model.

6 Qualitative Analysis

Entities as Existential Quantifiers: Table 5 shows the body of two horn clauses. Both the clauses are predictive of the fact $\text{location.contains}(x, b)$. The first clause is *universally* true irrespective of the entities present in the chain (transitive property). However the value of the second clause is only true *conditional* on the instantiations of the entities. The score of the Path-RNN model is independent of the entity values, whereas our model outputs a different score based on the entities in the chain. We average the scores across entities, which are connected through this path and for which the relation holds in column 3 (With Entities).

For the first clause, which is independent of entities, both models predict a high score. However for the second clause, the model without entity information predicts a lower score because this path is seen in both positive and negative training ex-

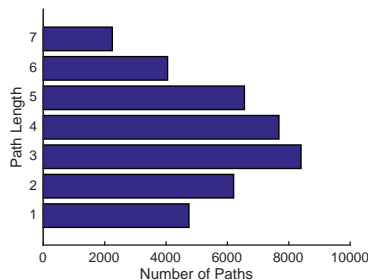


Figure 4: Length distribution of top-scoring paths

amples and the model cannot condition on the entities to learn to discriminate. However our model predicts the true relations with high confidence. This is a first step towards the capturing existential quantification for logical inference in vector space.

Length of Clauses: Figure 4 shows the length distribution of top scoring paths in the test set. The distribution peaks at lengths= $\{3, 4, 5\}$, suggesting that previous approaches (Yang et al., 2015) which restrict the length to 3 might limit performance and generalizability.

Limitation: A major limitation of our model is inability to handle long textual patterns because of sparsity. Compositional approaches for modeling text (Toutanova et al., 2015; Verga et al., 2016) are a right step in this direction and we leave this as future work.

7 Conclusion

This paper introduces a single high capacity RNN model which allows chains of reasoning across multiple relation types. It leverages information from the intermediate entities present in the path between an entity pair and mitigates the problem of unseen entities by representing them as a function of their annotated types. We also demonstrate that pooling evidence across multiple paths improves both training speed and accuracy. Finally, we also address the problem of reasoning about infrequently occurring relations and show significant performance gains via multitasking.

References

- Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *NAACL*.
- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.
- Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2014. Recursive neural networks for learning logical semantics. *CoRR*.
- Rich Caruana. 1997. Multitask learning. *Machine Learning*.
- Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*.
- Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M. Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *EMNLP*.
- Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *EMNLP*.
- Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. *Lexical and Computational Semantics*.
- K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, Stroudsburg, PA, USA.
- Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *EMNLP*.
- Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*.
- Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL*.
- Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*, Beijing, China.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.
- Dave Orr, Amar Subramanya, Evgeniy Gabrilovich, and Michael Ringgaard. 2013. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. <http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html>.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. *UAI '09*.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.
- Tim Rocktäschel and Sebastian Riedel. 2016. Learning knowledge base inference with neural theorem provers. In *AKBC, NAACL*.
- Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In *In Introduction to SRL*.
- Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *EMNLP*.
- Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *ACL*.
- Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *AKBC, CIKM*.
- Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.
- Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoi-fung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, September.
- Kristina Toutanova, Xi Victoria Lin, Scott Wen tau Yih, Hoi-fung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. *ACL*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *ICLR*.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Incorporating relation paths in neural relation extraction. *CoRR*.

Recognizing Mentions of Adverse Drug Reaction in Social Media Using Knowledge-Infused Recurrent Models

Gabriel Stanovsky*

Bar-Ilan University

Ramat Gan, Israel

gabriel.satanovsky@gmail.com

Daniel Gruhl

IBM Research Almaden

San Jose, California, USA

dgruhl@us.ibm.com

Pablo N. Mendes*

Lattice Data, Inc.

Menlo Park, California, USA

pablomendes@gmail.com

Abstract

Recognizing mentions of Adverse Drug Reactions (ADR) in social media is challenging: ADR mentions are context-dependent and include long, varied and unconventional descriptions as compared to more formal medical symptom terminology. We use the CADEC corpus to train a recurrent neural network (RNN) transducer, integrated with knowledge graph embeddings of DBpedia, and show the resulting model to be highly accurate (93.4 F1). Furthermore, even when lacking high quality expert annotations, we show that by employing an active learning technique and using purpose built annotation tools, we can train the RNN to perform well (83.9 F1).

1 Introduction

Identifying medical concepts in social media narratives is the task of recognizing certain phrases in the context of a user's post. Each phrase is also assigned a label from a set of predefined medical types. For instance, given the sentence “*Aspirin cured my terrible headache, but made me sleepy*”, the following medical concepts can be identified: “*Aspirin*” is identified as **Drug**, “*terrible headache*” as a **Symptom** and “*made me sleepy*” should be spotted as a **Adverse Drug Reaction**.

Having an automatic identification process can help domain experts examine large quantities of unstructured data, and quickly identify emerging trends. For example, associating previously unknown side effects with a given drug, or identifying an unforeseen impact to a change in the manufacturing process.

There are several challenges in addressing this task. First, *context* is crucial to type assignment. Compare the previous example with “*Aspirin cured my sleepiness but gave me a terrible headache*”, while the medical concepts are similar, their context determines their particularly associated type label.

The social media domain poses additional challenges. User narratives on social platforms tend to be non-grammatical, use colloquialisms, slang, and generally informal language. For example, a user may express sleepiness as “*hard time getting some Z's*”. This hinders the use of pre-trained statistical parsers or simple string matching techniques.

In this work we focus on the identification of *Adverse Drug Reactions* (ADR). These are unwanted side effects which the user clearly identifies as caused by the intake of a drug. ADRs are particularly challenging to spot, as they can be articulated in a variety of ways and can often be confounded with the symptoms addressed by the drug.

Previous work in this field has mainly used carefully built lexicons and hand-coded rule based systems (Iqbal et al., 2015). While each individual system achieves good results in the particular domain, porting these rules to another domain is non-trivial. For example, identifying psychiatric drug adverse reactions will probably consist of a much different lexicon than that of cardiac medication.

In this work we address ADR mention recognition by with recurrent neural network (RNN) transducers (Graves, 2012). We propose a framework which makes novel use of general, non-task-specific medical knowledge from DBpedia (Lehmann et al., 2015).

Our contributions are two fold: first, we use the high-quality annotation of the CSIRO Adverse Drug Event Corpus (CADEC) (Karimi et al., 2015) to train accurate models, achieving perfor-

* Work performed while at IBM Research Almaden.

mance of 93.4 F1 on the CADEC test section. Analyzing the performance of our models, we show that the theoretically unbounded memory of the RNN is good at capturing the context of the narratives, and that external DBpedia knowledge provides additional improvements.

Second, in most medical domains there is no large preexisting collection of expert annotation (i.e., gold standard test and training data) and obtaining one is an expensive prospect. We therefore address the following research question: “How quickly can our system converge on ‘good enough’ annotations?”.

For that end, we use only the test portion of CADEC to test a model trained on non-expert annotation. To expedite this process, we use a purpose built annotation tool in conjunction with an active learning technique to sample the most informative examples to annotate. This approach achieves reasonable results in a very short time (83.9 F1 in only one hour of human annotation). We suggest that this framework is a promising avenue for researches exploring low-resource domains, alleviating the need to first commit to an expensive annotation endeavor.

2 Background

In this section we describe the CADEC corpus, which we use to train and test our model, and DBpedia, along with the recent paradigm of knowledge graph embeddings integrated into our RNN.

2.1 CSIRO Adverse Drug Event Corpus

The recently created CSIRO Adverse Drug Event Corpus (CADEC) (Karimi et al., 2015) contains medical concepts annotation in posts from `Ask a Patient`¹, an online forum collecting medical patient narratives.

For example, a forum entry regarding a certain drug starts with “*I experienced one night of agonising upper stomach pain, diarrhoea and sleeplessness*”.

CADEC used `brat` (Stenetorp et al., 2012) to annotate five types of medical concepts: (1) *Drug*, names of medicine or drug, e.g., “*Diclofenac*” or “*Aspirin*”; (2) *Adverse Drug Reaction*, an unwanted reaction which according to the text is clearly associated with taking the drug, e.g., “*acute stomach pain*”; (3) *Disease*, the reason for taking the drug, e.g., “*insomnia*” or “*aggression*”;

¹<http://www.askapatient.com>

(4) *Symptom*, manifestations of the disease, e.g., “*trouble sleeping*” or “*constantly angry*”; and finally (5) *Finding*, a clinical finding that does not pertain to any of the above categories.

Each annotation consists of a word span (possibly non-contiguous) and a mapping of the marked span to medical ontologies (SNOMED (Cote et al., 1977), AMT², and MedDRA (Brown et al., 1999)).

Each post is annotated by either a medical student or a computer scientist, screened by the authors of the papers, and finally reviewed by a clinical terminologist. The annotations spanned 1,244 posts relating to 12 drugs divided into two groups (medications with Diclofenac as an active ingredient, and Lipitor). Corpus statistics are presented in Table 1.

In the course of this work we will use the CADEC Adverse Drug Reaction annotations to train and test our models.

2.2 DBpedia and Knowledge Graph Embeddings

DBpedia is a large-scale cross-domain multilingual knowledge base extracted from Wikipedia (Lehmann et al., 2015). DBpedia uses a schema with over 320 entity types and 1,600 property types to describe nearly 4 million entities. Besides the common Person, Location and Organization entity types, it also includes descriptions of drugs, diseases, symptoms and disorders, among others.

Using a knowledge graph such as DBpedia requires an intimate knowledge of its entity and relation types, as well as its subtle representation decisions. This creates challenges with using knowledge graphs in a machine learning (ML) setting, where the signals are coming from different sources and are often normalized and assimilated to make the final prediction.

²<https://goo.gl/xRCGPN>

	Train	Test	All
# Posts	935	309	1244
# Sentences	5723	1874	7597
# Words	95979	31855	127834
# Unique Words	5788	3373	9161

Table 1: Statistics for the CADEC corpus (See section 2).

In order to allow ML algorithms to make use of the information encapsulated in such graphs, a recent line of work (Bordes et al., 2011; Bordes et al., 2013; Wang et al., 2014; Ji et al., 2016) has compellingly suggested to embed entities as d dimensional dense real vectors, and relations as two projection matrices: R^{lhs} and R^{rhs} . Similarly to word embedding techniques (Collobert et al., 2011; Mikolov et al., 2013), a deep learning model is trained to differentiate between observed (positive) and non-observed (negative) triples, by minimizing the following score for positive triples (E_i, R, E_j):

$$\|R^{lhs}E_i - R^{rhs}E_j\| \quad (1)$$

During training, the model learns the entity and relation embeddings, desirably encoding some of the semantics and co-occurrence information of the original knowledge graph.

3 Recognizing Mentions of Adverse Drug Reaction

In this section we formally define the task of in-context recognition of Adverse Drug Reactions (ADR) mentions and describe our proposed problem modeling.

3.1 Task Formulation

We follow CADEC’s definition for ADR, as described in Section 2.

Formally, we define the ADR mention recognition task as a sentence level chunking task, where each word can either be: (1) Beginning of an ADR span (B); (2) Inside an ADR span (I); or (3) Outside of the span of an ADR (O);

For example (tags in subscript):

“ I_o stopped $_o$ taking $_o$ Ambien $_o$ after $_o$ three $_o$ weeks $_o$ – i_o gave $_o$ me $_o$ a $_o$ terrible $_B$ headache $_I$ ”

This formulation, termed BIO tagging (Ramshaw and Marcus, 1995; Sang and Veenstra, 1999), is equivalent to noun phrase (NP) chunking annotation convention with a single type of NP.

While Ratinov and Roth (2009) have shown that a more elaborate tagging scheme (BILOU)³ improved performance in their experiments in Named Entity Recognition, those experiments are out of the scope for this work.

³BILOU uses tags for: Beginning, Inside, Last and Unit length chunks.

This task depends heavily upon context, as the same word span can appear as an ADR in one text, and as a Symptom in another. For example, the first entry in Table 2 mentions several ADRs (“made me gain 30 lbs”, “made my BP go up so high”, “gave me more anxiety”) associating each with a different drug (“Klonopin”, “Lexapro”, etc.), while the second entry in the table uses some of the same surface forms to refer to an addressed Symptoms (e.g., “It helped both my anxiety and IBS”).

Furthermore, our model will need to cope with texts from social media which tend to be colloquial, non-grammatical, variably spelled and overall employ highly informal phrasing. The rest of the entries in Table 2 present several snippets from the Ask a Patient corpus, illustrating some of these challenges.

3.2 Recurrent Neural Networks Transducer

Formulated this way, and given that the sentences in Ask a Patient are of arbitrary length, it seems applicable to model the task using Recurrent Neural Networks (RNNs). This approach was proven to be effective in many recent NLP papers. For a recent and extensive survey of RNNs in NLP see (Goldberg, 2015).

Specifically, we use a bi-directional LSTM transducer (Graves, 2012) which outputs a probability distribution over the three possible labels (B, I, and O) per word, taking into account arbitrary length contexts from both past as well as future words.

Pretrained word embeddings It is common in recent neural networks frameworks to initialize the model’s word embeddings with pretrained parameters, from a much larger (often unsupervised) corpus. We experiment with initializing our word embeddings from both out of domain (and out of the box) word embeddings from Google (Mikolov et al., 2013), as well as with purpose trained embeddings utilizing predicate-argument structure from Open-IE (Etzioni et al., 2008) (following (Stanovsky et al., 2015)) from the Blekko medical corpus (a 2GB corpus of web pages categorized as “medical domain” by the Blekko search engine⁴).

4 Augmenting RNNs with DBpedia

Despite the original motivation for knowledge graph embedding, few efforts were made to use

⁴<https://en.wikipedia.org/wiki/Blekko>

Text Snippet	Challenges
"I've tried Klonopin which gave me nightmarish side effects, Lexapro which made me gain 30 lbs and that gave me more anxiety and borderline depression, Effexor which made my BP go up so high I was hospitalized for 4 days (high bp runs in the family), and Buspar which did n't even touch my anxiety."	Long post describing Adverse Drug Reactions caused by various previous drugs.
"It helped both my anxiety and IBS immensely."	Describes cure from symptoms (not ADRs).
"After the second pill the same progression of symptoms only now the abdominal gas, cramps and pain would be with me all day."	Coordination leads to non-contiguous ADR spans ("abdominal cramps", "abdominal pain").
"I had the usual problems as most people. Driving, buying things online, cooking, eating, sexual activity."	The qualifier "while asleep" is implied by "as most people" but never explicitly stated.
"Short term more loss"	Ungrammatical.
"started having tension headaches. did not relate to Ambien."	The first sentence implies ADRs, while the second negates them.

Table 2: Examples from the Ask a Patient forum.

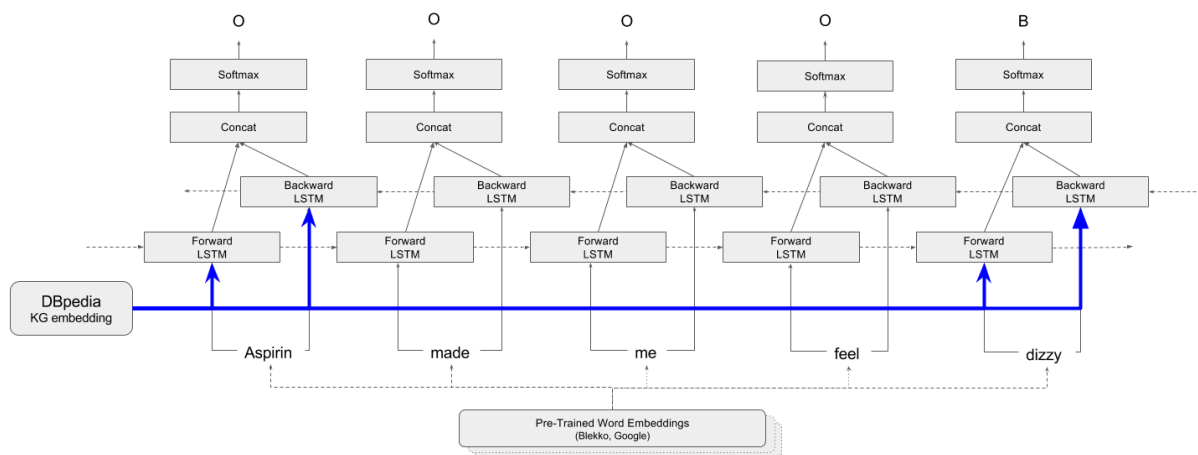


Figure 1: The bi-LSTM transducer, integrated with DBpedia knowledge graph embedding (left). We experiment with several corpora for training external word embeddings (bottom) and override them for DBpedia concepts (e.g., “Aspirin”, “dizzy”). See Sections 3 and 4 for details.

such embeddings as components in larger NLP frameworks. Instead, previous research has focused on embedding techniques, as outlined in section 2. In this section we describe a novel framework which utilizes DBpedia concepts embeddings, in addition to the common use of pre-trained word embeddings. We specifically use DBpedia due to its good coverage of our domain of interest. Furthermore, since it relies on Wikipedia, it might also be applicable for non-medical domains. The presented approach, however, is not limited to any particular knowledge base and in future work we plan to extend it beyond DBpedia.

The motivation for using external knowledge bases in our case stems from the relatively small size of the CADEC corpus (see Table 1), in comparison with other neural models training corpora. For example, The Penn Treebank (Marcus et al., 1993) which is often used for training dependency parsing algorithms, consists of roughly 7M tokens, versus only about 95K tokens in CADEC.

4.1 Overriding Word Embeddings with DBpedia Concepts

We augment our model with a pretrained knowledge graph embedding of the “Drug” and “Disease” categories from DBpedia, training as discussed in Section 2.2. When a word in a CADEC entry is a lexical match with one of the DBpedia entities we override its features with the DBpedia embeddings. Intuitively, this framework introduces complex semantic relations between prominent and task relevant words in the Ask a Patient posts. For example, DBpedia draws “Aspirin” and “Ibuprofen” closer in the embedding space as both appear under the “Non-steroidal anti-inflammatory drug” category (a relation which is modeled in DBpedia). While this changes the embedding of a small subset of the words, these are meaningful and frequently occurring in our setting (see details in Section 6).

Figure 1 shows the complete architecture of our model, including the RNN transducer LSTM and the pretrained word embeddings augmented with DBpedia entity embeddings. The loss from the network propagates back to the word embeddings, allowing them to assimilate task-specific information during training.

5 Human in the Loop

From our experience, real world applications often do not have a pre-existing rich gold standard corpus from which they can efficiently train high quality models. This lack creates a serious impediment to entering and exploring the opportunities for text analytics in such domains, due to the high cost of producing the requisite semantic assets.

The medical domain is one where this is especially true. Even within a particular specialty, e.g., oncology, very different information may need to be extracted depending on the type of cancer being explored. In domains where the patient’s verbatim comments are critical (e.g., psychiatric, physical rehabilitation) there can be even more variability and ambiguity. For instance, a comment of feeling “pins and needles” could be a result of peripheral neurological issues, or a panic attack, and may be expressed with multiple misspellings, punctuation and grammar variations.

To that end, we test our suggested model in a human-in-the-loop approach to gauge how quickly an analytic developer might obtain “good enough” training and test data to develop first generation code and begin to explore the results.

To simulate this for an experiment, we ignore the CADEC training annotations and instead interleave adjudication of small batches (100 sentences) with iterations of model training (see Figure 2). To allow for these fast iterations we need to provide solutions in several areas. For quicker annotation, we developed RASCAL (Rapid Adjudication of Semantic Classes to Accelerate Learning), a purpose built annotation tool which expedites the annotation and adjudication process. Furthermore we employ an active learning technique to focus the human adjudicator’s time on examples that the model finds most confusing.

This process, as elaborated below, has two outcomes: (1) Expedited machine-assisted production of an annotated gold standard; and (2) Rapid training of high precision models due to the active learning technique.

5.1 Bootstrap

While the learning process can be initiated by simply beginning to annotate the corpus, we find a more rapid start up is achieved by employing an extensive lexicon of Adverse Drug Reaction phrases. Fortunately, lexicon expansion techniques (Codon et al., 2012) provide a way to

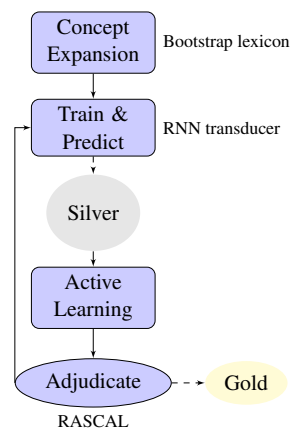


Figure 2: Train-Predict-Adjudicate loop

rapidly bootstrap this portion of the problem. Typically the user provides a few (~ 3) examples of the type desired, and the system comes back with dozens of suggestions of potential new phrases. The user approves or rejects these and the system repeats with this additional knowledge. The process typically generates a couple of hundred candidate terms in a few minutes.

5.2 Active Learning

The bootstrap lexicon can then be used to obtain a preliminary noisy CADEC training set for the RNN, by marking each occurrence of a lexicon term as an ADR. After we train our model, we want to choose informative samples to adjudicate and refine our training set (and subsequently, our model) in the next iteration.

A process in which a model chooses its next training examples is often referred to as *active learning*, and is a well researched area of machine learning (see (Settles, 2010) for an extensive survey). For our purposes, we use the uncertainty sampling criterion (Lewis and Gale, 1994). Intuitively, this ranks the samples according to the model’s belief it will mislabel them.

Formally, we sort all samples (x, y) according to the following measure:

$$1 - \Pr_{\theta}(\hat{y}|x) \quad (2)$$

Where:

$$\hat{y} = \operatorname{argmax}_{y \in \{B, I, O\}} \Pr_{\theta}(y|x) \quad (3)$$

We choose the top 100 samples according to this metric, and adjudicate them, as described below. In order to assess the impact of this step we also perform the same process with random sampling of 100 sentences at each iteration.

5.3 Adjudication

At this stage, a human adjudicator examines the sentences chosen in the previous phase. They then either accept or reject each automatically recognized ADR span in these sentences. Additionally the annotator can mark new spans that were missed.

While the `brat` annotator (Stenetorp et al., 2012) is a popular tool for creating and modifying annotations, it is a bit cumbersome and error prone for tasks such as the one outlined in CADEC.

Notably, the CADEC creators mention that the `brat` annotations required an additional cleaning phase. For example `brat` annotates character spans instead of aligning marked spans to the word level (e.g. an annotator might wrongfully mark “*pai*” instead of “*pain*”). While some of `brat`’s more complex features are a good fit for other annotation tasks, they reduce agility and do not add much value in ours. We therefore implemented a simple rapid adjudication system (code-named RASCAL) that is tuned to the particular task of adjudicating and adding annotations in the context of ADR mention recognition.

RASCAL introduces simplifications such as single click removal of incorrect annotations, automatic alignment of the spans to include whole tokens, and single key “approve and move to next document” support. This results in very fast annotation times. As the system improves its understanding of the entity to tag, much of the annotator’s time is spent simply approving annotations, with about one in four requiring their addition of a missed span.

Over the 1,100 annotations we found an average time of about 3 seconds per sentence. In a controlled experiment comparing the annotation of 100 sentences in `brat` versus RASCAL, this represented at least a four fold improvement over comparable `brat` times without pre-annotation. Since we find `brat` is slower at removing errors and entering split annotations we anticipate the discrepancy may be even higher with pre-annotation.

This improvement does have a cost, however; RASCAL only allows a single annotation type at a time, so the annotation of two predefined types (e.g., Drug and ADR) requires two passes. Second, RASCAL does not support non-contiguous span annotations. This is of especial trouble when there are coordinated spans (e.g., “*my neck and*

back are both spasming” should be “*neck spasming*” and “*back spasming*”).

While these were uncommon in our corpus (see detailed analysis in Section 6), it does suggest that perhaps performing some of the annotations with `brat` after doing the initial ones with RASCAL might help improve precision if desired.

5.4 Repeat

Given these adjudicated annotations, we can refine our bootstrap lexicon (with the newly acquired ADR mentions) and automatically re-annotate the entire training corpus according to it, generating a new iteration of the training data which closes the loop back to training and predicting (Section 5.2).

Knowing when to stop is always a challenge with learning systems. For the sake of these experiments we chose to stop after an hour of an annotator time (the initial lexicon expansion bootstrap and annotating/adjudicating 1,100 sentences).

However the human annotator using RASCAL gets a fairly good sense of what kind of annotations are being spotted and what is being missed. By looking at the net change in the pre-annotation to post-annotation spans for an iteration it is possible to get a sense of when the learning is leveling off.

In future work, it may also be possible to look at the total uncertainty the RNN finds in the training corpus before and after a training session as a measure of how much more productive learning there may be left.

6 Evaluation

In this section we evaluate our recurrent neural transducer in two scenarios: (1) Using the high quality annotations of the CADEC corpus and (2) Simulating a task with low resources in an active learning scenario and using RASCAL for non-expert annotations, as described in Section 5. Results are shown in Table 3.

Label imbalance The CADEC corpus is imbalanced between the different labels, assigning the label “O” (Outside) to 87.34% of the words. This is due to the fact that most of the text in the `Ask a Patient` forum describes background situation and is not directly related to an ADR (see, for example, the first entry in Table 2). This poses a problem for training accuracy oriented models, as such imbalanced class distribution discourages the learning process to move from a model which

assigns a constantly higher probability to the Outside label, regardless of the input sentence (He and Garcia, 2009). Subsequently, this leads to trivial solutions which achieve 0% ADR recall (as no ADRs are retrieved) and 100% ADR precision (as there are also no false positives).

To address this problem we use the SMOTE (Synthetic Minority Over-Sampling) technique (Chawla et al., 2002) which skews the sample distribution by oversampling the minority classes (B and I) during training to get a synthetically balanced training set.

6.1 Experimental setup

We implemented the bi-LSTM transducer model using the Keras framework (Chollet, 2015) with a TensorFlow backend (Abadi et al., 2015). Open IE word embeddings (300 dimensions) were trained on Blekko medical corpus (1 billion tokens) using Open IE 4⁵ and Word2Vec (Mikolov et al., 2013), as described in (Stanovsky et al., 2015). For DBpedia embeddings (300 dimensions), we used the code published in (Nickel et al., 2015). We used the code published in (Lemaître et al., 2016) for SMOTE class resampling. Finally, we used the `libact` library (Yang et al., 2015) for the active learning sampling. All models were trained for 100 epochs.

6.2 Results

Several observations can be made based on the results of our experiments (Table 3):

RASCAL achieves good results at a fraction of the annotation effort - RASCAL results are obtained after just 11 cycles of annotation by single annotator (roughly an hour of work), and are then tested against the independently annotated test set of CADEC. The performance of RASCAL is a promising indication that adequately performing models can be obtained very quickly using our framework, when moving to a new annotation task where training data is scarce.

External knowledge improves performance in both scenarios - As can be seen from the ablation test in Table 3, in both supervised and annotator development settings, our pretrained embeddings improve performance by at least 13 points

⁵<https://github.com/allenai/openie-standalone>

in F1, with a significant edge to Blekko embeddings. This is in part due to its better coverage of the CADEC lexicon, only 408 (7.05% of the CADEC lexicon) unique words were Out Of Vocabulary (OOV) using Blekko, compared with 724 (12.51%) OOV words using Google’s embedding. DBpedia provides embeddings for 232 words (4%) and further adds 2-4 points in both recall and precision.

Uncertainty sampling boosts the learning rate

- Figure 3 shows the progression of the best performance obtained at each training iteration. The uncertainty sampling (see Section 5.2) boosts the learning curve, achieving models performing around 80 F1 after just 25 minutes of RASCAL annotation (note the red vertical dotted line).

The relatively small change when increasing the number of annotated instances from 400 to 800 (i.e., before and after the vertical dotted line) is probably due to the long tail nature of the problem: active learning chooses the most prominent examples first, then there is a sharp decline in the novelty of the chosen examples. Further experimentation with active learning techniques may improve performance, yet this falls out of the scope of this paper, and is left as a topic for future research. Overall, it can be seen from Figure 3 that our active learning technique is indeed already superior to random sampling (notice the brown dotted line indicating performance with random sampling).

Context matters - We tested an oracle ADR baseline which had access to the lexicon of all of the ADRs in CADEC. This oracle ignored context and marked every occurrence of a phrase from the lexicon as an ADR. As can be seen in Table 3 (ADR Oracle), this baseline obviously achieves 100% recall, yet, more interestingly, it achieves only 55.2% in precision. Thus in 44.8% of the cases the surrounding context negated the ADR phrase (for example, see the last entry in Table 2).

6.3 Error analysis

In analyzing the RASCAL model, we find that it relatively lacks in recall. This is due to our limited annotation effort having predictably limited coverage. Examining our annotations, we find 449 unique ADRs annotated in RASCAL out of the total 3685 unique ADR phrases in the full CADEC annotation. The RNN model is in fact able to generalize these mentions and find approximately

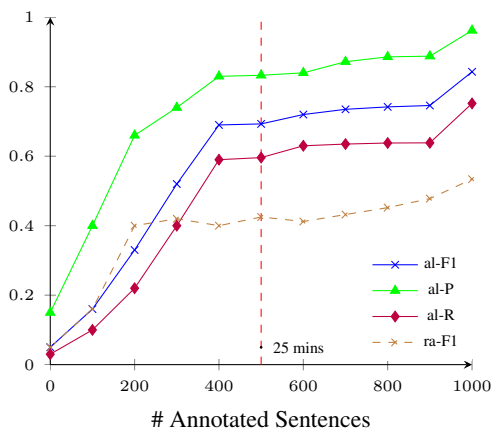


Figure 3: Model performance per annotation cycle (see Section 6). Solid lines represent precision, recall and F1 for active learning, and the dashed line represents F1 for random sampling (precision and recall follow the same trend and are omitted for clarity).

	P	R	F1
String matching	78.6	37.0	50.3
RASCAL	64.0	53.8	58.0
RASCAL+ Google	88.4	68.2	76.8
RASCAL+ Blekko	91.3	70.0	79.4
RASCAL+ DBpedia+ Blekko	96.2	75.2	83.9
ADR Oracle	55.2	100	71.1
CADEC	69.6	74.6	71.9
CADEC + Google	85.3	86.2	85.7
CADEC + Blekko	90.5	90.1	90.3
CADEC + DBpedia+ Blekko	92.2	94.5	93.4

Table 3: Performance of the different baselines by training from RASCAL annotations (top) vs. CADEC training data (bottom). See Section 6 for more details.

75% of the mentions, yet it is likely that having a larger RASCAL training set would help improve the coverage of our model. Furthermore, the design choices made in RASCAL trade annotation speed with accuracy. As mentioned in Section 5.3 RASCAL is currently unable to annotate non-contiguous spans, which account for 1005 (15.9%) of the ADRs annotated in CADEC.

Finally, both of our models predict BIO word labels at the sentence level which in some cases does not provide enough context to arrive at the correct label. See, for example, the bottom example in Table 2, in which a very probable ADR phrase in the first sentence (“*tension headaches*”) is negated in the second sentence (“*did not relate to Ambien*”).

7 Related Work

To the best of our knowledge, there has been no previous work attempting to recognize in-context adverse drug reaction mentions on the CADEC corpus. There are, however, several papers which addressed the same task on a different corpus, and others who have used the CADEC corpus for orthogonal tasks. In this section we survey two such recent papers.

Limsopatham and Collier (2016) have used CADEC for the normalization of medical concepts. They take as input an out-of-context ADR (e.g., “*I couldn’t sleep all night*” or “*head explodes*”) and predict its normalized form (e.g., “*insomnia*” or “*headache*”, respectively), based on a predefined vocabulary. They use an RNN model and report accuracy of 79.98. This task can be seen a subsequent task to ours. The ADR spans we output can serve as an input for ADR normalization, giving medical experts a consolidated summary of the reported adverse events.

Iqbal et al. (2015) share our motivation to identify ADR mentions in the context of electronic health records (medical correspondence, discharge letters, etc.), which are more formal, as opposed to our focus on social media domain. They take a rule based approach, and come up with an expert built lexicon, which achieves 85 F1 on their test set.

While their approach is carefully built to the specific data set, we show the portability of our model by testing both in a supervised scenario as well as in annotation development scenario.

8 Conclusions and Future Work

We presented a novel model which consists of an LSTM transducer RNN augmented with external knowledge from medically oriented Web crawl and a knowledge graph embedding of medical entities in DBpedia. We showed that the model achieves good results (93.4 F1) when trained and tested on the CADEC corpus.

Furthermore, ignoring the CADEC training data, we showed that through active learning and a task-dedicated annotation tool we can get a reasonably performing model (83.95 F1 on CADEC’s test set) with just an hour of annotation effort. This suggests a promising methodology for researchers wanting to explore new domain annotations, without first committing to a heavyweight and expensive annotation effort.

Future work may make further use of the CADEC annotations (e.g., for multi-task learning or concept normalization), and extend RASCAL to get better recall and allow for non-contiguous and multiple label annotations.

Acknowledgments

We would like to thank the IBM Almaden Intelligence Augmentation Team (Alfredo Alba, Linda Kato, Chris Kau, Joe Terdiman and Steve Welch) for many fruitful discussions, and the anonymous reviewers for their helpful comments.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.
- Elliot G. Brown, Louise Wood, and Sue Wood. 1999. The medical dictionary for regulatory activities (meddra). *Drug Safety*, 20(2):109–117.
- Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.
- François Chollet. 2015. keras. <https://github.com/fchollet/keras>.
- Anni Coden, Daniel Gruhl, Neal Lewis, Michael Tanenblatt, and Joe Terdiman. 2012. Spot the drug! an unsupervised pattern matching method to extract drug names from very large clinical corpora. In *Healthcare Informatics, Imaging and Systems Biology (HISB), 2012 IEEE Second International Conference on*, pages 33–39. IEEE.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Roger A. Cote, College of American Pathologists, et al. 1977. *Systematized nomenclature of medicine*. College of American Pathologists.
- Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the Web. *Communications of the ACM*, 51(12):68–74.
- Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.
- Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.
- Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.
- Ehtesham Iqbal, Robbie Mallah, Richard George Jackson, Michael Ball, Zina M Ibrahim, Matthew Broadbent, Olubanke Dzahini, Robert Stewart, Caroline Johnston, and Richard JB Dobson. 2015. Identification of adverse drug events from free text electronic patient records and information in a large mental health case register. *PLoS one*, 10(8):e0134208.
- Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of AAAI*.
- Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81.
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.
- Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2016. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *CoRR*, abs/1609.06570.
- David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.
- Nut Limsopatham and Nigel Collier. 2016. Normalising medical concepts in social media texts by learning semantic representation. ACL.

- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2015. Holographic embeddings of knowledge graphs. *arXiv preprint arXiv:1510.04935*.
- Lance A Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. *arXiv preprint cmp-lg/9505040*.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.
- Erik F. Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 173–179. Association for Computational Linguistics.
- Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.
- Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer.
- Yao-Yuan Yang, Yu-An Chung, Shao-Chuan Lee, Tung-En Wu, and Hsuan-Tien Lin. 2015. libact: Pool-based active learning in python. Technical report.

Multitask Learning for Mental Health Conditions with Limited Social Media Data

Adrian Benton
Johns Hopkins University
adrian@cs.jhu.edu

Margaret Mitchell
Microsoft Research¹
mmitchellai@google.com

Dirk Hovy
University of Copenhagen
mail@dirkhovy.com

Abstract

Language contains information about the author’s demographic attributes as well as their mental state, and has been successfully leveraged in NLP to predict either one alone. However, demographic attributes and mental states also interact with each other, and we are the first to demonstrate how to use them jointly to improve the prediction of mental health conditions across the board. We model the different conditions as tasks in a multitask learning (MTL) framework, and establish for the first time the potential of deep learning in the prediction of mental health from online user-generated text. The framework we propose significantly improves over all baselines and single-task models for predicting mental health conditions, with particularly significant gains for conditions with limited data. In addition, our best MTL model can predict the presence of conditions (neuroatypicality) more generally, further reducing the error of the strong feed-forward baseline.

1 Introduction

Mental health conditions, like depression or anxiety, are still one of the leading causes of death worldwide. Suicide, often the direct outcome of mental health conditions, is the 11th most frequent cause of death in the US (Anderson, 2001). Detecting mental health risk factors early is key to preventing many of these deaths. Unfortunately, traditional diagnosis methods require access to and willingness to talk with a psychologist, and rely mainly on impressions formed during short

sessions. Consequently, conditions leading to preventable suicides can often not be accurately diagnosed.

Automated monitoring and risk assessment of patients’ language have the potential to overcome the logistic and time constraints associated with traditional assessment methods. Written text carries implicit information about the author, a relationship that has been exploited in natural language processing (NLP) to predict author characteristics, such as age (Goswami et al., 2009; Rosenthal and McKeown, 2011; Nguyen et al., 2011; Nguyen et al., 2014), gender (Sarawgi et al., 2011; Ciot et al., 2013; Liu and Ruths, 2013; Alowibdi et al., 2013; Volkova et al., 2015; Hovy, 2015), personality and stance (Schwartz et al., 2013b; Schwartz et al., 2013a; Volkova et al., 2014; Plank and Hovy, 2015; Park et al., 2015; Preotiuc-Pietro et al., 2015), or occupation (Preotiuc-Pietro et al., 2015a; Preotiuc-Pietro et al., 2015b). The same signal has also been effectively used to predict mental health conditions, such as depression (Coppersmith et al., 2015b; Schwartz et al., 2014), suicidal ideation (Coppersmith et al., 2016; Huang et al., 2015), schizophrenia (Mitchell et al., 2015) or post-traumatic stress disorder (PTSD) (Pedersen, 2015), often more accurately than by traditional diagnoses.

However, these studies typically model each condition in isolation and ignore other author attributes that can improve prediction, thereby artificially limiting performance. Existing research, however, indicates that 1) incorporating demographic attributes can help text classification (Volkova et al., 2013; Hovy, 2015), and 2) learning several auxiliary tasks which share common structures (e.g., part-of-speech tagging, parsing, and NER) can improve performance, as the learning implicitly exploits interactions between the tasks (Caruana, 1993; Sutton et al., 2007; Rush et al.,

¹Now at Google Research.

2010; Collobert et al., 2011; Søgaaard and Goldberg, 2016).

In this paper, we propose such a multitask learning (MTL) approach to mental health prediction. The main tasks of our model are predictions of *neurotypicality* (i.e., the absence of any mental health conditions), *anxiety*, *depression*, *suicide attempt*, *eating disorder*, *panic attacks*, *schizophrenia*, *bipolar disorder*, and *post-traumatic stress disorder (PTSD)*. All of the above, plus gender prediction, also serve as auxiliary tasks.

The auxiliary tasks reflect the observation that several conditions frequently occur together (comorbidity), and that they correlate with demographic factors. The MTL framework allows us to share information across predictions. We use a neural architecture that enables the inclusion of several loss functions with a common shared underlying representation. This experimental setup is flexible enough to extend this model to further factors than the ones shown here, provided suitable data.

We also explore the effect of auxiliary-task selection on model performance for a given prediction task. Similar to Caruana (1996), we find that choosing auxiliary tasks which are prerequisites or related to the main task is critical for learning a strong model.

Our contributions

1. We are the first to apply MTL to predict mental health conditions from user content on Twitter – a notoriously difficult task (Coppersmith et al., 2015a; Coppersmith et al., 2015b).
2. We explore the influence of auxiliary-task selection on prediction performance, including the effect of gender
3. We show how to model tasks with a *large* number of positive examples to improve the prediction accuracy of tasks with a *small* number of positive examples.
4. We increase the True Positive Rate at 10% false alarms by up to 9.7% absolute (for anxiety), a result with direct impact for clinical applications.

2 Model Architecture

We opt for a neural architecture to exploit the synergies between mental conditions. Our choice is based on practical more than ideological reasons: previous work (Collobert et al., 2011; Caruana,

1996; Caruana, 1993) has indicated that this is a promising model architecture, which allows us to share parameters across tasks, can be trained on large amounts of data, and accounts for varying degrees of annotation across tasks.¹

Even within the neural model framework, however, there are many variations to consider. In the following, we outline some attributes and decisions.

Previous approaches have shown considerable improvements over single task models by using MTL (Caruana, 1993). The arguments are convincing: predicting multiple related tasks should allow us to exploit any correlations between the predictions.

However, we note that the benefit of using a MTL model is only one possible explanation, and that another, more salient factor might have been overlooked: the difference in the general model class, i.e., neural architectures vs. discriminative or generative models, or, more generally, the *expressivity* of the model. Some comparisons might therefore have inadvertently compared apples to oranges.

We compare the multitask demographics and risk prediction with models with equal expressivity. We evaluate the performance of a standard logistic regression model (a standard approach to text-classification problems), a multilayer perceptron single-task learning (STL) model, and a neural MTL model, the latter two with equal numbers of parameters. This ensures a fair comparison by isolating the unique properties of MTL from the dimensionality-reduction aspects of deep architectures in general.

The neural models we evaluate come in two forms. The first, depicted in plate notation in Figure 1, is the STL model. These are feedforward networks with two hidden layers, trained independently to predict each task. On the right of Figure 1 is the MTL model, where the first hidden layer from the bottom is shared between all tasks. An additional per-task hidden layer is used to give the model flexibility to map from the task-agnostic representation to a task-specific one. Each hidden layer uses a rectified linear unit as non-linearity. The output layer uses a logistic non-linearity, since all tasks are binary predictions.

¹We also experimented with a graphical model architecture, but found that it did not scale as well and provided less traction.

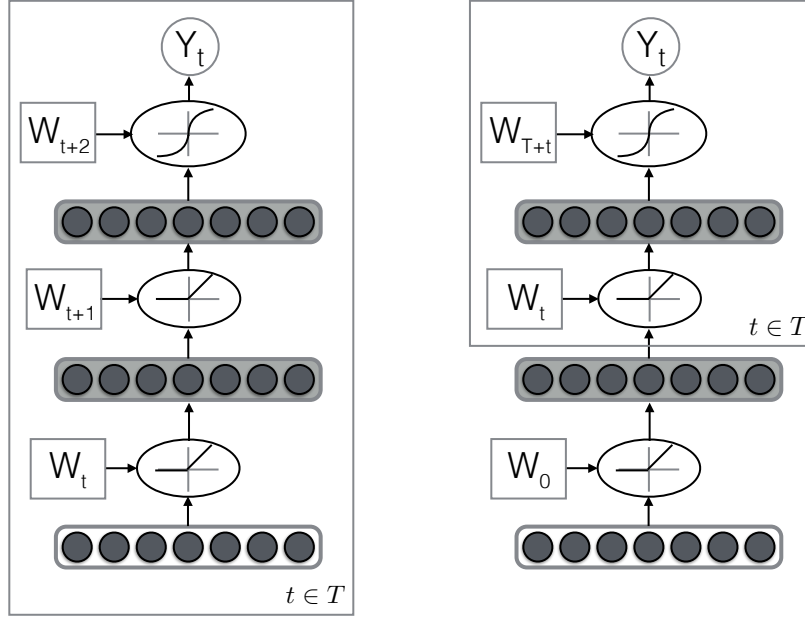


Figure 1: STL model in plate notation (left): weights trained independently for each task t (e.g., anxiety, depression) of the T tasks. MTL model (right): shared weights trained jointly for all tasks, with task-specific hidden layers.

Curves in ovals represent the type of activation used at each layer (rectified linear unit or sigmoid). Hidden layers are shaded.

The MTL model can easily be extended to a stack of shared hidden layers, allowing for a more complicated mapping from input to shared space.²

As noted in (Collobert et al., 2011), MTL benefits from mini-batch training, which both allows optimization to jump out of poor local optima, and take more stochastic gradient steps in a fixed amount of time (Bottou, 2012). In that paper, the authors use a randomized selection over the tasks to train. In our paper, we create mini-batches by sampling from the users in our data. Each of these users has some subset of the mental conditions we are trying to predict, and may or may not be annotated with gender. At each mini-batch gradient step we update weights for all tasks. This not only allows for randomization and faster convergence, it also provides a speed-up over the individual selection process in (Collobert et al., 2011).

One of the advantages of our setup is that we do not need complete information for every instance: instead, learning can proceed with asynchronous updates dependent on what the data in each batch

has been annotated for, while sharing representations throughout. This effectively learns a joint model with a common representation for several different tasks, and allows the use of several “disjoint” data sets, some with limited annotated instances.

Optimization and Model Selection Even in a relatively simple neural model, there are a number of parameters that can (and have to) be tuned to achieve good performance. We perform a line search for every model we use, sweeping over L_2 regularization and hidden layer width. We select the best model based on the development loss. Figure 5 shows the performance on the corresponding test sets (plot smoothed by rolling mean of 10 for visibility).

In our experiments, we sweep over the L_2 regularization constant applied to all weights in $\{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.5, 1.0, 5.0, 10.0\}$, and hidden layer width (same for all layers in the network) in $\{16, 32, 64, 128, 256, 512, 1024, 2048\}$. We fix the mini-batch size to 256, and 0.05 dropout on the input layer. We found that choosing a small mini-batch size and the model with low-

²We tried training a 4-shared-layer MTL model to predict targets on a separate dataset, but did not see any gains over the standard 1-shared-layer MTL model in our application.

Task	# of users
Neurotypicality	4820
Anxiety	2407
Depression	1400
Suicide attempt	1208
Eating disorder	749
Schizophrenia	349
Panic disorder	263
Bipolar disorder	234
PTSD	191
Female Male	788 248
total	9611

Table 1: Number of users with each self-stated condition and human-annotated gender in the joined dataset.

est development loss was sufficient to account for overfitting.

We train each model for 5,000 iterations, jointly updating all weights in our models. After this initial joint training, we select each task separately, and only update the task-specific layers of weights independently for another 1,000 iterations (selecting the set of weights achieving lowest development loss for each task individually). Weights are updated using mini-batch Adagrad (Duchi et al., 2011) – we found this to converge more quickly than other optimization schemes we experimented with. We evaluate the tuning loss every 10 epochs, and evaluate the model with the lowest tuning loss.

3 Data

We train our models on a union of multiple Twitter user datasets: 1) users identified as having anxiety, bipolar disorder, depression, panic disorder, eating disorder, PTSD, or schizophrenia (Coppersmith et al., 2015a), 2) those who had attempted suicide (Coppersmith et al., 2015c), and 3) those identified as having either depression or PTSD from the 2015 Computational Linguistics and Clinical Psychology Workshop shared task (Coppersmith et al., 2015b), along with neurotypical gender-matched controls (Twitter users not identified as having a mental condition). Users were identified as having one of these conditions if they stated explicitly they were diagnosed with this condition on Twitter (verified by a human annotator). For a subset of 1,101 users, we also have manually-annotated gender. The final dataset contains 9,611 users in total, with an average of 3521 tweets per

user. The number of users with each condition is included in Table 1. Users in this joined dataset may be tagged with multiple conditions, thus the counts in this table do not sum to the total number of users.

We use the entire Twitter history of each user as input to the model, and split it into character 1-to-5-grams, which have been shown to capture more information than words for many Twitter text classification tasks (McNamee and Mayfield, 2004; Coppersmith et al., 2015a). We compute the relative frequency of the 5,000 most frequent n -gram features for $n \in \{1, 2, 3, 4, 5\}$ in our data, and then feed this as input to all models. This input representation is common to all models, allowing for fair comparison.

4 Experiments

Our task is to predict any number of mental conditions for each of the users in these data, possibly using gender prediction as an auxiliary task to improve our prediction performance.

We evaluate three classes of models: a baseline logistic regression over character n -gram features (LR), feed-forward multilayer perceptrons trained to predict each task separately (STL), and a multi-task network predicting a set of conditions simultaneously (MTL). We also perform ablation experiments, to see which subsets of auxiliary tasks help us learn an MTL model that predicts a particular mental condition best. For all experiments, data were divided into five equal-sized folds, three for training, one for tuning, and one for testing (we report the performance on this).

All our models are implemented in Keras³ with Theano backend and GPU support. We train the models for a total of up to 15,000 epochs, using mini-batches of 256 instances. Training time on all five training folds ranged from one to eight hours on a machine with Tesla K40M.

Evaluation Setup We compare the accuracy of each model at predicting each task separately.

In clinical settings, we are interested in minimizing the number of false positives, i.e., incorrect diagnoses, which can cause undue stress to the patient. We are thus interested in bounding this quantity. To evaluate the performance, we plot the false positive rate (FPR) against the true positive rate (TPR). This gives us a receiver operating characteristics (ROC) curve, allowing us to inspect the

³<http://keras.io/>

performance of each model on a specific task at any level of FPR.

While the ROC gives us a sense of how well a model performs at a fixed true positive rate, it makes it difficult to compare the individual tasks at a low false positive rate, which is also important for clinical application. We therefore report two more measures: the area under the ROC curve (AUC) and TPR performance at FPR=0.1 (TPR@FPR=0.1). We do not compare our models to a majority baseline model, since this model would achieve an expected AUC of 0.5 for all tasks, and F-score and TPR@FPR=0.1 of 0 for all mental conditions – users exhibiting a condition are the minority, meaning a majority baseline classifier would achieve zero recall.

5 Results

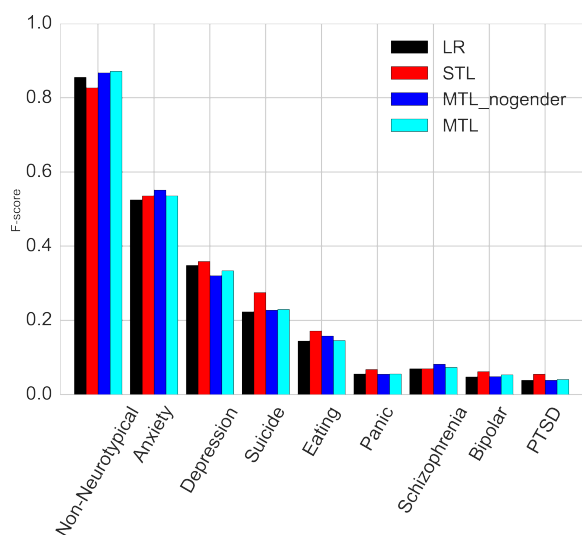


Figure 2: F1-score for predicting each condition.

Figure 2 shows the F1-score of each model at predicting each task separately, Figure 3 shows the AUC-score, and Figure 4. Precision-recall curves for each of model/task are in Figure 6.

STL corresponds to a multilayer perceptron with two hidden layers (with a similar number of parameters as the proposed MTL model). The MTL_nogender and MTL models predict all tasks simultaneously, but are only evaluated on the main respective task.

MTL often outperforms the LR baseline in terms of AUC and TPR@F=0.1, but the difference is less clear when comparing F1-scores.

In terms of AUC and TPR@F=0.1, STL models do not perform nearly as well as MTL or LR. This is likely because the neural networks learned by

STL cannot be guided by the inductive bias provided by MTL training. Note, however, that STL and MTL are oftentimes comparable in terms of F1-score.

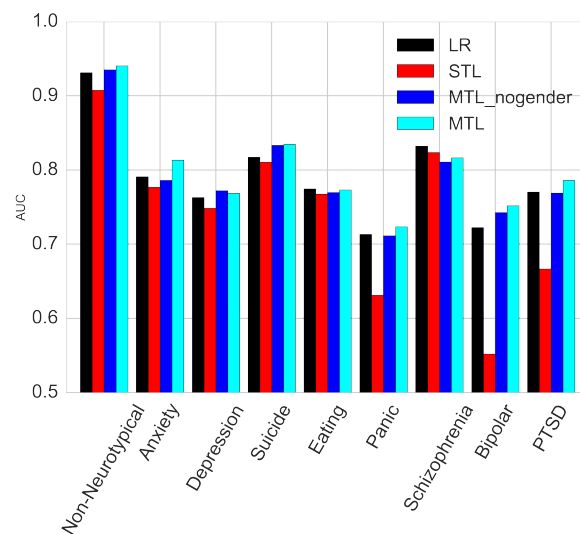


Figure 3: AUC for predicting different tasks

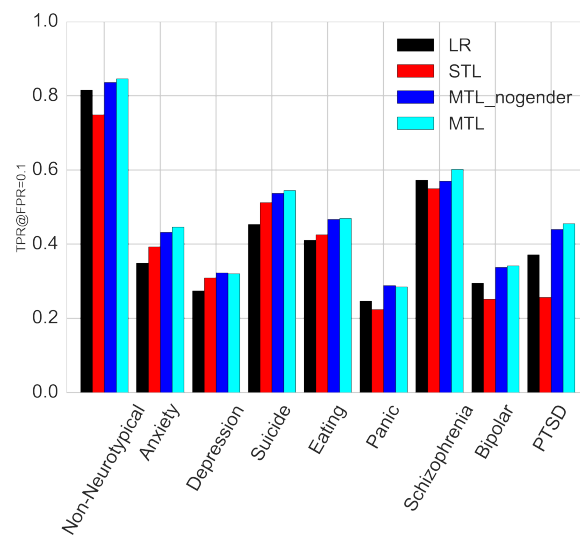


Figure 4: TPR at 0.10 FPR for predicting different tasks

MTL Leveraging Comorbid Conditions Improves Prediction Accuracy

We find that the prediction of the conditions with the least amount of data – *bipolar disorder* and *PTSD* – are significantly improved by forcing the model to also predict comorbid conditions which have substantially more data: *depression* and *anxiety*. We are able to increase the AUC for predicting PTSD to 0.786 by MTL, from 0.770 by LR, whereas STL fails to perform as well with an AUC of

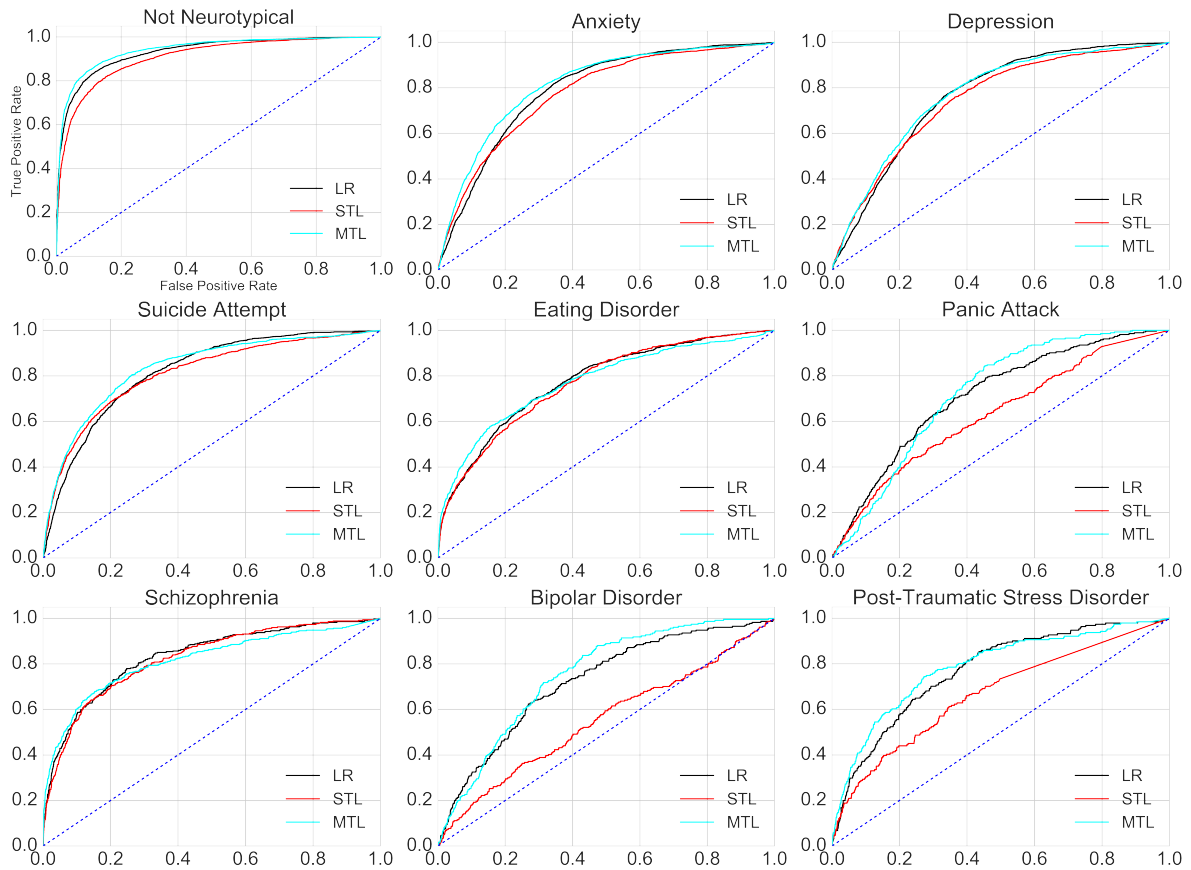


Figure 5: ROC curves for predicting each condition. The precision (diagnosed, correctly labeled) is on the y -axis, while the proportion of false alarms (control users mislabeled as diagnosed) is on the x -axis. Chance performance is indicated by the dotted diagonal line.

0.667. Similarly for predicting bipolar disorder (MTL:0.723, LR:0.752, STL:0.552) and panic attack (MTL:0.724, LR:0.713, STL:0.631).

These differences in AUC are significant at $p = 0.05$ according to bootstrap sampling tests with 5000 samples. The wide difference between MTL and STL can be explained in part by the increased feature set size – MTL training may, in this case, provide a form of regularization that STL cannot exploit. Further, modeling the common mental health conditions with the most data (depression, anxiety) helps in pulling out more rare conditions that also manifest in these common health conditions.

This is clear evidence that an MTL model provides strong gains for predicting elusive conditions by using large data for common conditions, and only a small amount of data for the related, small-data conditions.

Utility of Authorship Attributes Figures 3 and 4 both suggest that adding gender as an auxiliary task leads to more predictive models, even

though the difference is not statistically significant for most tasks. This is in line with the suggestions in Volkova et al. (2013), Hovy (2015). Interestingly, though, the MTL model is worse at predicting gender itself. While this could be a direct result of data sparsity (recall that we have only a small subset annotated for gender), which could be remedied by annotating additional users for gender, this appears unlikely given the other findings of our experiments, where MTL helped in specifically these sparse scenarios.

However, it has been pointed out by Caruana (1996) that not all tasks benefit from a MTL setting in the same way, and that some tasks serve purely auxiliary functions. Here, gender prediction does not benefit from including mental conditions, but helps vice versa. In other words, predicting gender is qualitatively different from predicting mental health conditions: it seems likely that the signals for anxiety are much more similar to the ones for depression than for, say, being male, and can therefore add to detecting de-

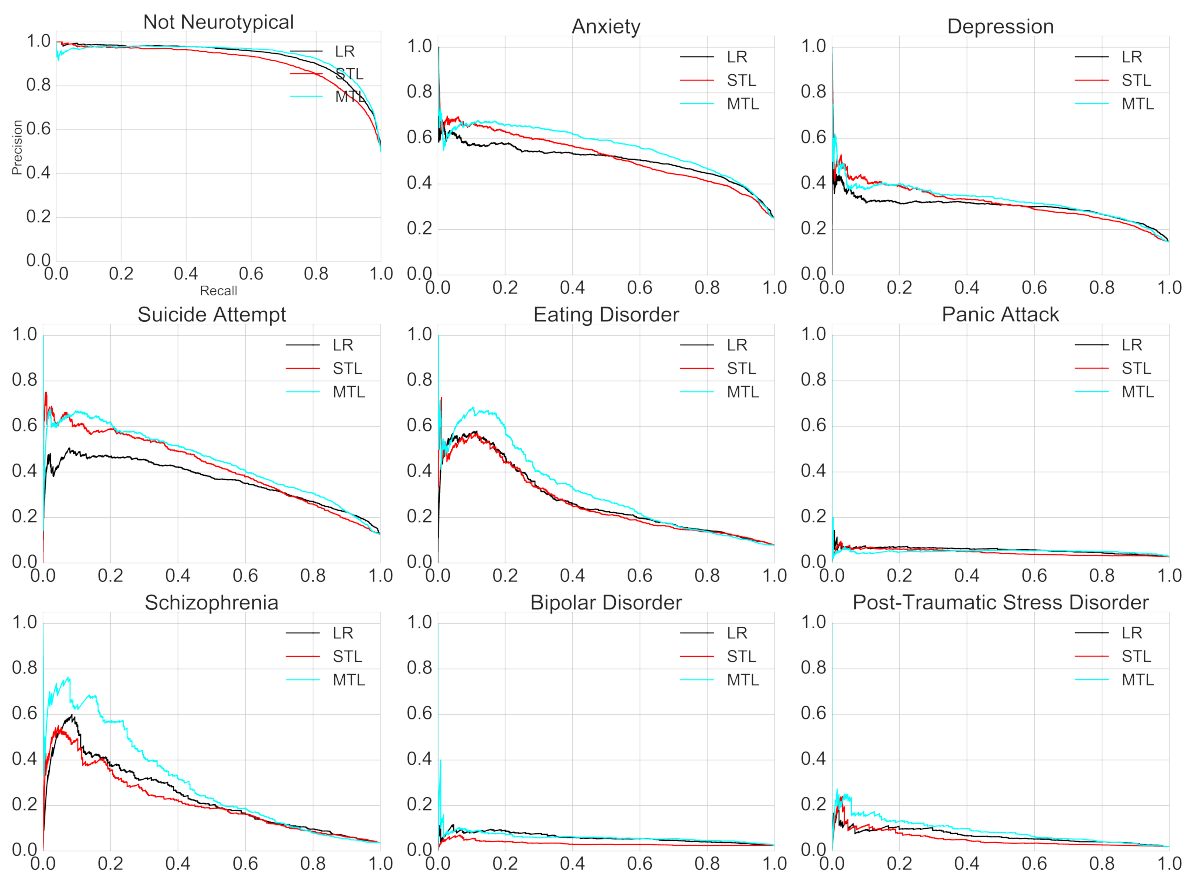


Figure 6: Precision-recall curves for predicting each condition.

pression. However, the distinction between certain conditions does not add information for the distinction of gender. The effect may also be due to the fact that these data were constructed with inferred gender (used to match controls), so there might be a degree of noise in the data.

Choosing Auxiliary Tasks Although MTL tends to dominate STL in our experiments, it is not clear whether auxiliary tasks just introduce a beneficial bias in MTL models in general, or if there exists a specific subset of auxiliary tasks for predicting each condition. We perform ablation experiments by training MTL models on only a subset of the tasks, and evaluate them at predicting a single target. We focus on four conditions we want to predict well: anxiety, depression, suicide attempts, and bipolar disorder. For each task, we vary the auxiliary tasks we train the MTL model with, and evaluate how well it predicts the main task. Since considering all possible subsets of tasks as auxiliary tasks is combinatorily unfeasible, we choose the following task subsets as auxiliary:

- *all*: all mental conditions along with gender

- *all conds*: only all mental conditions (gender omitted)
- *neuro*: only neurotypicality
- *neuro+mood*: neurotypicality, depression, and bipolar disorder (mood disorders)
- *neuro+anx*: neurotypicality, anxiety, and panic attack (anxiety conditions)
- *neuro+targets*: neurotypicality, anxiety, depression, suicide attempt, and bipolar disorder
- *none*: no auxiliary tasks, equivalent to STL model

Table 2 shows AUC for the four prediction tasks with different subsets of auxiliary tasks. Statistically significant improvements over the respective LR baselines are denoted by superscript. Restricting the auxiliary tasks to a small subset tends to hurt performance for most tasks. This suggests that the biases induced by predicting any mental condition are all mutually beneficial – e.g., models that predict depression, are also useful at predicting anxiety.

It is thus best not to think of MTL as one single “black box” model that can predict all mental con-

Auxiliary Tasks	Main Task			
	anxiety	bipolar	depression	suicide attempt
<i>all</i>	0.813 ^{*†}	0.752 ^{*†}	0.769 [†]	0.835 ^{*†}
<i>all conds</i>	0.786	0.743 [†]	0.772 [†]	0.833 ^{*†}
<i>neuro</i>	0.763	0.740 [†]	0.759	0.797
<i>neuro+mood</i>	0.756	0.742 [†]	0.761	0.804
<i>neuro+anx</i>	0.770	0.744 [†]	0.746	0.792
<i>neuro+targets</i>	0.750	0.747 [†]	0.764	0.817
<i>none (STL)</i>	0.777	0.552	0.749	0.810
<i>LR</i>	0.791	0.723 [†]	0.763	0.817

Table 2: Test AUC when predicting *Main Task* after training to predict a subset of auxiliary tasks. Significant improvement over LR baseline at $p = 0.05$ is denoted by ^{*}, and over no auxiliary tasks (STL) by [†].

ditions at the same time, but a framework to exploit auxiliary tasks as regularization to effectively combat data paucity and less-than-trustworthy labels.

6 Discussion

Our results indicate that the proposed MTL setting results in significant gains for the prediction of mental health conditions with limited data, benefiting from predicting related mental conditions and demographic attributes simultaneously.

We experimented with all the optimizers that Keras provides, and found that Adagrad seems to converge fastest to a good optimum, although all the adaptive learning rate optimizers (such as Adam, etc.) tend to converge quickly. This indicates that the gradient is significantly steeper along certain parameters than others. Default stochastic gradient descent (SGD) was not able to converge as quickly, since it is not able to adaptively scale the learning rate for each parameter in the model – taking too small steps in directions where the gradient is shallow, and too large steps where the gradient is steep. We further note an interesting behavior: all of the adaptive learning rate optimizers yield a strange “step-wise” training loss learning curve, which hits a plateau, but then drops after about 900 iterations, only to hit another plateau, and so on. Obviously, we would prefer to have a smooth training loss curve. We can indeed achieve this using SGD, but it takes much longer to converge than, for example, Adagrad. This suggests that a well-tuned SGD would be the best optimizer

Learning Rate	Loss	L2	Loss	Hidden Width	Loss
10^{-4}	5.1	10^{-3}	2.8	32	3.0
$5 * 10^{-4}$	2.9	$5 * 10^{-3}$	2.8	64	3.0
10^{-3}	2.9	10^{-2}	2.9	128	2.9
$5 * 10^{-3}$	2.4	$5 * 10^{-2}$	3.1	256	2.9
10^{-2}	2.3	0.1	3.4	512	3.0
$5 * 10^{-2}$	2.2	0.5	4.6	1024	3.0
0.1	20.2	1.0	4.9		

Table 3: Average dev loss over epochs 990-1000 of joint training on all tasks as a function of different learning parameters. Optimized using Adagrad with hidden layer width 256.

for this problem, a step that would require some more experimentation and is left for future work.

We also found that feature counts have a pronounced effect on the loss curves: relative feature frequencies yield models that are much easier to train than raw feature counts.

As indicated by the effect of raw vs. relative counts, feature representations are another area of optimization, such as different ranges of character n -grams (e.g., $n > 5$) and word unigrams. We decided on character 1-to-5-grams, since we believe that these features generalize better to a new domain (e.g., Facebook) than word unigrams. However, there is no fundamental reason not to choose longer character n -grams, other than time constraints in regenerating the data, and sufficiently accounting for overfitting with proper regularization.

Initialization is often listed as a decisive factor in neural models, and Goldberg (2015) recommends repeated restarts with differing initializations to find the optimal model. In an earlier experiment, we tried initializing a MTL model (albeit without task-specific hidden layers) with pre-trained word2vec embeddings of unigrams trained on the Google News n -gram corpus. However, we did not notice an improvement in F-score. This could be due to the other factors, though, such as feature sparsity.

Table 3 displays sweeps over learning parameters with hidden layer width 256, training the MTL model to predict multiple mental conditions jointly for the Qntfy self-stated data (character tri-grams as input features). The sweet spots in this table are probably good starting points for training models.

7 Related Work

MTL was introduced by Caruana (1993), based on the observation that humans rarely learn things in isolation, and that it is the similarity between related tasks that helps us get better.

Some of the first works on MTL have been motivated by medical risk prediction (Caruana et al., 1996), and it is now being rediscovered for this purpose (Lipton et al., 2016). The latter use a long short-term memory (LSTM) structure to provide several medical diagnoses from health care features (yet no textual or demographic information), and find small, but probably not significant improvements over a structure similar to the STL we use here.

However, in both cases, the target was medical conditions as detected in patient records, not mental health conditions in online data. The main focus in this work has been on the correlation between individual conditions and linguistic markers, to establish the possibility of detecting risk in written data. While some of the approaches have looked at more than one condition, none of them have done so in an MTL framework, foregoing the possibility of modeling comorbidity and correlation with demographic factors.

The framework proposed by Collobert et al. (2011) allows for predicting any number of NLP tasks from a convolutional neural network (CNN) representation of the input text. The model we present is much simpler, just a feed-forward network with n -gram input layer. Our contribution is to show that constraining n -gram embeddings to be predictive of various mental health condition also helps. We chose to experiment with a feed-forward network against independent logistic regression models since this was the simplest way to test our hypothesis. Comparing more complicated models is possible, but distracts from the question whether or not MTL training with extra-linguistic targets helps us.

8 Conclusion and Future Work

In this paper, we develop neural MTL models for 10 prediction tasks (eight mental health conditions, neurotypicality, and gender). We compare their performance with STL models trained to predict each task independently.

Our results show that the most complex MTL model performs significantly better than independent LR models, reaching 0.846 TPR where

FPR=0.1 and reducing the error rate in identifying anxiety by up to 11.9%. We also investigate the influence of the depth of the model, by comparing to progressively deeper STL feed-forward networks with the same number of parameters. We find: (1) Most of the modeling power stems from the expressiveness conveyed by deep architectures. (2) Choosing the correct set of auxiliary tasks for a given mental condition can yield a significantly more predictive model. (3) The MTL model dramatically improves for conditions with the smallest amount of data. (4) Gender prediction does not follow the two previous points, but improves performance when added as an auxiliary task.

Accuracy of the MTL approach is not yet ready to be used in isolation in the clinical setting. However, our experiments suggest this is a promising direction moving forward. There are strong gains to be made in using multitask learning to aid clinicians in their evaluations, and with further partnerships between the clinical and machine learning community, we foresee improved suicide prevention efforts.

In the future, we plan to explore the possibility of hierarchical models, encoding the fact that certain tasks inform others more than vice versa.

Acknowledgements

The authors would like to thank Kristy Hollingshead Seitz, Glen Coppersmith, and H. Andrew Schwartz, as well as the organizers and funders of the Johns Hopkins Jelinek Summer School 2016, where large parts of this work were conducted. We are also grateful for the invaluable feedback on MTL from Yoav Goldberg, Stephan Gouws, Ed Greffentette, Karl Moritz Hermann, and Anders Sjøgaard.

References

- Jalal S Alowibdi, Ugo A Buy, and Philip Yu. 2013. Empirical evaluation of profile characteristics for gender classification on twitter. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 1, pages 365–369. IEEE.
- Robert N. Anderson. 2001. *Deaths: leading causes for 1999*. Centers for Disease Control and Prevention, National Center for Health Statistics.
- Léon Bottou. 2012. Stochastic gradient tricks. *Neural Networks, Tricks of the Trade, Reloaded*, pages 430–445.

- Rich Caruana, Shumeet Baluja, Tom Mitchell, et al. 1996. Using the future to “sort out” the present: Rankprop and multitask learning for medical risk evaluation. *Advances in neural information processing systems*, pages 959–965.
- Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Citeseer.
- Rich Caruana. 1996. Algorithms and applications for multitask learning. In *ICML*, pages 87–95. Citeseer.
- Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender Inference of Twitter Users in Non-English Contexts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Wash.*, pages 18–21.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. 2015a. From adhd to sad: Analyzing the language of mental health on twitter through self-reported diagnoses. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 1–10.
- Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, and Margaret Mitchell. 2015b. Proceedings of the 2nd workshop on computational linguistics and clinical psychology: From linguistic signal to clinical reality. pages 31–39. Association for Computational Linguistics.
- Glen Coppersmith, Ryan Leary, Eric Whyne, and Tony Wood. 2015c. Quantifying suicidal ideation via language usage on social media. In *Joint Statistics Meetings Proceedings, Statistical Computing Section, JSM*.
- Glen Coppersmith, Kim Ngo, Ryan Leary, and Anthony Wood. 2016. Proceedings of the third workshop on computational linguistics and clinical psychology. pages 106–117. Association for Computational Linguistics.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.
- Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.
- Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. 2009. Stylometric analysis of bloggers’ age and gender. In *Third International AAAI Conference on Weblogs and Social Media*.
- Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of ACL*, pages 752–762.
- Xiaolei Huang, Xin Li, Tianli Liu, David Chiu, Ting-shao Zhu, and Lei Zhang. 2015. Topic model for identifying suicidal ideation in chinese microblog. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 553–562.
- Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. 2016. Learning to diagnose with lstm recurrent neural networks. In *Proceedings of ICLR*.
- Wendy Liu and Derek Ruths. 2013. What’s in a name? Using first names as features for gender inference in Twitter. In *Analyzing Microtext: 2013 AAAI Spring Symposium*.
- Paul McNamee and James Mayfield. 2004. Character n-gram tokenization for european language text retrieval. *Information retrieval*, 7(1-2):73–97.
- Margaret Mitchell, Kristy Hollingshead, and Glen Coppersmith. 2015. Quantifying the language of schizophrenia in social media. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 11–20, Denver, Colorado, June 5. Association for Computational Linguistics.
- Dong Nguyen, Noah A Smith, and Carolyn P Rosé. 2011. Author age prediction from text using linear regression. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 115–123. Association for Computational Linguistics.
- Dong Nguyen, Dolf Trieschnigg, A. Seza Dogruöz, Rilana Gravel, Mariet Theune, Theo Meder, and Franciska De Jong. 2014. Predicting Author Gender and Age from Tweets: Sociolinguistic Theories and Crowd Wisdom. In *Proceedings of COLING 2014*.
- Greg Park, H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, David J Stillwell, Michal Kosinski, Lyle H Ungar, and Martin EP Seligman. 2015. Automatic personality assessment through social media language. *Journal of Personality and Social Psychology*.
- Ted Pedersen. 2015. Proceedings of the 2nd workshop on computational linguistics and clinical psychology: From linguistic signal to clinical reality. pages 46–53. Association for Computational Linguistics.
- Barbara Plank and Dirk Hovy. 2015. Personality traits on twitter—or—how to get 1,500 personality tests in a week. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 92–98.

- Daniel Preoțiu-Pietro, Johannes Eichstaedt, Gregory Park, Maarten Sap, Laura Smith, Victoria Tobolsky, Hansen Andrew Schwartz, and Lyle H Ungar. 2015. The role of personality, age and gender in tweeting about mental illnesses. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, NAACL.
- Daniel Preoțiu-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015a. An analysis of the user occupational class through twitter content. In *ACL*.
- Daniel Preoțiu-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying user income through language, behaviour and affect in social media. *PloS one*, 10(9):e0138717.
- Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre-and post-social media generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 763–772. Association for Computational Linguistics.
- Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.
- Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. 2011. Gender attribution: tracing stylometric evidence beyond topic and genre. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 78–86. Association for Computational Linguistics.
- Hansen Andrew Schwartz, Johannes C Eichstaedt, Lukasz Dziurzynski, Margaret L Kern, Eduardo Blanco, Michal Kosinski, David Stillwell, Martin EP Seligman, and Lyle H Ungar. 2013a. Toward personality insights from language exploration in social media. In *AAAI Spring Symposium: Analyzing Microtext*.
- Hansen Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013b. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9).
- Andrew H. Schwartz, Johannes Eichstaedt, L. Margaret Kern, Gregory Park, Maarten Sap, David Stillwell, Michal Kosinski, and Lyle Ungar. 2014. Proceedings of the workshop on computational linguistics and clinical psychology: From linguistic signal to clinical reality. pages 118–125. Association for Computational Linguistics.
- Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 231.
- Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.
- Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of EMNLP*, pages 1815–1827.
- Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proceedings of the 52nd annual meeting of the ACL*, pages 186–196.
- Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media (demo). In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI)*, Austin, TX, January.

Evaluation by Association: A Systematic Study of Quantitative Word Association Evaluation

Ivan Vulić¹, Douwe Kiela², and Anna Korhonen¹

¹ Language Technology Lab, DTAL, University of Cambridge

² Facebook AI Research

{iv250|alk23}@cam.ac.uk dkiela@fb.com

Abstract

Recent work on evaluating representation learning architectures in NLP has established a need for evaluation protocols based on subconscious cognitive measures rather than manually tailored intrinsic similarity and relatedness tasks. In this work, we propose a novel evaluation framework that enables large-scale evaluation of such architectures in the free word association (WA) task, which is firmly grounded in cognitive theories of human semantic representation. This evaluation is facilitated by the existence of large manually constructed repositories of word association data. In this paper, we (1) present a detailed analysis of the new quantitative WA evaluation protocol, (2) suggest new evaluation metrics for the WA task inspired by its direct analogy with information retrieval problems, (3) evaluate various state-of-the-art representation models on this task, and (4) discuss the relationship between WA and prior evaluations of semantic representation with well-known similarity and relatedness evaluation sets. We have made the WA evaluation toolkit publicly available.

1 Introduction

The quality of word representations in semantic models is often measured using intrinsic evaluations that capture particular types of relationships (typically semantic similarity and relatedness) between word pairs (Finkelstein et al., 2002; Hill et al., 2015; Schnabel et al., 2015; Tsvetkov et al., 2015, *inter alia*).

Whereas the notions of semantic *similarity* and *relatedness* constitute key concepts in such evaluations, they are in fact vaguely defined (Batchkarov

et al., 2016; Ettinger and Linzen, 2016). The construction of ground truth evaluation sets that reflect these relations, such as SimLex-999 (Hill et al., 2015), SimVerb-3500 (Gerz et al., 2016), MEN (Bruni et al., 2014) or Rare Words (Luong et al., 2013), relies on manually constructed guidelines that trigger subjective human interpretation of the task at hand. This in turn introduces inter-annotator variability (Batchkarov et al., 2016) and does not account for the fact that human similarity judgments are asymmetric by nature (Tversky, 1977).

What is more, given that humans perform linguistic comparisons between concepts on a subconscious level (Kutas and Federmeier, 2011), it is at least debatable whether current similarity/relatedness evaluation sets fully capture the implicit relational structure underlying human language representation and understanding.

As evidenced by recent workshops on evaluation of semantic representations¹, the community appears to recognise that current evaluation methods are inadequate. To fill in this gap, recent work has proposed using subconscious cognitive measures of semantic connection instead, as a proxy for measuring the ability of statistical models to tackle various problems in human language understanding (Ettinger and Linzen, 2016; Søgaard, 2016; Mandera et al., 2017).

Motivated by these insights, this work proposes an evaluation framework based on the word association (WA) task, firmly rooted in and described by the psychology literature, e.g., Nelson et al. (2000) and Griffiths et al. (2007)². Word associations, provided as simple (*cue, response*) concept pairs, are naturally asymmetric: they tend to be given as a repository of ranked lists of concepts col-

¹E.g. RepEval, <https://sites.google.com/site/repevalacl16/>

²The WA task is a free-association task, in which participants are asked to produce the first word that came into their head in *response* to a *cue* or *query* word.

lected as responses (i.e., associations) given a target cue/query concept. The ranking of the response list is based on the WA strength between the cue and each generated response. WAs are directly tied to language use and the memory systems that support online linguistic processing (Till et al., 1988; Nelson et al., 1998).

We build our WA evaluation framework around a large repository of the University of South Florida (USF) association norms (Nelson et al., 2000; Nelson et al., 2004). After post-processing, the repository contains ~5K queries, and ~70,000 (*cue, response*) pairs, making it one of the largest semantic evaluation databases available (by contrast, the largest word pair scoring data sets in NLP, SimVerb and MEN, contain 3,500 and 3,000 word pairs respectively). This new resource enables comprehensive quantitative studies of WA and may be used to guide the future development of representation learning architectures.

While parts of the USF data set have been used for evaluation in NLP before (Michelbacher et al., 2007; Silberer and Lapata, 2012; Kiela et al., 2014; Hill and Korhonen, 2014, *inter alia*), we conduct the first full study regarding the evaluation on the quantitative WA task. We compare a wide variety of different semantic representation models, discuss various evaluation metrics and analyse the links between word association and semantic similarity and relatedness. In summary, the main contributions of this paper are as follows:³

(C1) We present an end-to-end evaluation framework for the WA task, and provide new evaluation metrics and detailed guidelines for evaluating semantic models on the WA task.

(C2) We conduct a systematic study and comparison of current state-of-the-art representation learning architectures on the WA task.

(C3) We present a systematic quantitative analysis of the connections between the models' performance on the subconscious WA task and their performance on benchmarking similarity and relatedness evaluation sets.

2 Motivation: Association and USF

Implicit Cognitive Measures: Means of Semantic Evaluation? Several studies have shown clear correspondence between implicit cognitive

³All evaluation scripts and detailed evaluation guidelines are freely available at:
<https://github.com/cambridge/tl/wa-eval/>

measures (most notably *semantic priming*) and semantic relations encountered in vector space models (VSMs) (McDonald and Brew, 2004; Jones et al., 2006; Padó and Lapata, 2007; Herdağdelen et al., 2009), suggesting that some of the implicit relation structure in the human brain is already reflected in current statistical models of meaning.

These findings encouraged Ettinger and Linzen (2016) to propose a preliminary evaluation framework based on *semantic priming* experiments (Meyer and Schvaneveldt, 1971).⁴ They demonstrate the feasibility of such an evaluation using a subconscious language processing task. They use the online database of the Semantic Priming Project (SPP), which compiles priming data for over 6,000 word pairs.

Here, we go one step further and demonstrate that another subconscious language processing task, with much more available data, can also be used to evaluate representations. We construct an evaluation framework based on the USF free word association (WA) norms quantifying the strength of association between cue and response concepts for more than 70,000 concept pairs.

Word Association WA has been a long-standing research topic in cognitive psychology, as evidenced by the following statement (Deese, 1966):

Are there any more fascinating data in psychology than tables of association? (Deese, 1966)

Word association still remains one of the fundamental questions in cognitive psychology, as emphasised by e.g. Griffiths et al. (2007):

Association has been part of the theoretical armory of cognitive psychologists since Thomas Hobbes used the notion to account for the structure of our “trayne of thoughts” in 1651.

These insights illustrate how WA can provide a useful benchmark for evaluating models of human semantic representation. WA norms are commonly used in constructing memory experiments (Dennis and Humphreys, 2001; Steyvers and Malmberg, 2003), and statistics derived from them have been shown to be important in predicting cued recall

⁴Semantic priming measures a response time with a human subject performing a simple language task (e.g., classifying strings into words vs. non-words). It was shown that human subjects are able to solve the task more quickly if the word to which they are responding is preceded by a semantically related word. The magnitude of the speed-up can be taken as the strength of relation between the two concepts.

CUE	RESP	#G	#P	FSG	BSG
lunch	dinner	156	42	0.269	0.096
lunch	food	156	32	0.205	0.011
lunch	eat	156	13	0.083	0.0
lunch	meal	156	10	0.064	0.063
lunch	box	156	9	0.058	0.0
lunch	sandwich	156	9	0.058	0.037
lunch	noon	156	6	0.038	0.200
noon	lunch	150	30	0.200	0.038
noon	twelve	150	22	0.147	0.034
noon	sunshine	150	20	0.133	0.0
food	eat	180	73	0.406	0.409
food	drink	180	9	0.050	0.0

Table 1: Example (*cue, response*) pairs of free word association from the USF data set. #G stands for the number of participants serving in the group norming the word, while #P denotes the number participants producing a particular response.

and recognition (Nelson et al., 1998), and false memories (Roediger et al., 2001).⁵

WA Evaluation Set: USF The USF norms data set (hereafter **USF**) is the largest database of free word association collected for English (Nelson et al., 2004). It was generated by presenting human subjects with one of 5,000 cue concepts and asking them to write the first word coming to their mind that is associated with that concept. Each cue concept was normed by at least 100 participants, resulting in a set of associates (or *responses*) for each cue, for a total of $\sim 72,000$ (*cue, response*) pairs. A sample of the USF data is presented in Tab. 1. The data are accessible online.⁶

For each such pair, the proportion of participants that produced the response w^r when presented with cue word w^c can be used as a proxy for the strength of association between the two words (FSG in Tab. 1). BSG denotes the backward association strength, when the roles of a cue and a response are reversed, shows that the WA relation is inherently asymmetrical.

⁵From another viewpoint, the WA evaluation aims to answer a different question than a typical intrinsic evaluation on data sets such as SimLex-999, MEN, WordSim-353, or SimVerb-3500. The goal of the latter is to assess the quality of learned text representations as a proxy towards downstream NLP tasks. The goal of the former is to assess the capability of representation learning and NLP architectures to help in advancing our understanding and modeling of human cognitive processes (occurring on a sub-conscious level), while at the same time it could still be used as a proxy evaluation in NLP.

⁶<http://w3.usf.edu/FreeAssociation/>

3 Evaluation Protocol

Terminology $\mathcal{W}^c = \{w_1^c, \dots, w_i^c, \dots, w_{|\mathcal{W}^c|}^c\}$ denotes a set of $|\mathcal{W}^c|$ *cue* or *normed* words (more generally, concepts) in the evaluation set. For each cue word w_i^c , the data set contains a *ranked list* of concepts or *responses* \mathcal{R}_i sorted according to the strength of forward association, from cue to response (i.e., the FSG field in Tab. 1). The list \mathcal{R}_i contains entries of the format $w_{r,j} : \text{fsg}_{i,j}$, where $w_{r,j}$ is the j^{th} most associated concept in the ranked list, and $\text{fsg}_{i,j}$ is the accompanying strength of forward association between cue w_i^c and response $w_{r,j}$. Let \mathcal{R}_i^g refer to the ground truth ranked list for w_i^c , which contains only responses where $\text{fsg}_{i,j} > 0$ in the USF data, and \mathcal{R}_i^s to the ranked list retrieved by an automatic system.

The vocabulary or search space from which responses for all cues are drawn is labeled V^r . Note that V^r may also contain words from \mathcal{W}^c and that V^r may contain words that do not occur in any of the ground truth lists \mathcal{R}_i^g .

Why Evaluate on Word Association? A standard evaluation protocol with word pair scoring evaluation sets such as SimLex-999 or MEN is to compute Spearman’s ρ correlations between the ranking obtained by an automatic system and the ground truth ranking. This protocol, however, is not directly applicable to the USF test data. First, the evaluated relation of WA is *asymmetric*, and the pairs (X, Y) and (Y, X) may differ dramatically in their WA scores (see the difference in FSG and BSG values from Tab. 1). Second, instead of one global list of pairs, the data comprises a series of ranked lists conditioned on the cue/normed word w^c (see Tab. 1 again). Finally, unlike with SimLex-999 or MEN scores where it is difficult to interpret “what a similarity/relatedness of 7.69 exactly means” (Batchkarov et al., 2016; Avraham and Goldberg, 2016), the USF FSG scores have a direct meaningful interpretation (i.e., $FSG = \#P/\#G$). To fully capture all aspects of the ground truth USF data set, an evaluation protocol should ideally be based not only on response rankings, but also on the actual scores, i.e., the association strength.

In this paper, we propose and investigate two different families of evaluation metrics on the USF data: Sect. 3.1 discusses rank correlation evaluation metrics inspired by recent work on the evaluation of vector space models in distributional semantics (Bruni et al., 2014; Hill et al., 2015; Vulić et al.,

2016, inter alia). Sect. 3.2 draws inspiration from research on evaluation in information retrieval (IR). We show that the problem of evaluating USF association lists may be naturally framed as an ad-hoc IR task (Manning et al., 2008). This enables the application of standard IR evaluation methodology.

3.1 Rank Correlation Evaluation

Averaged Standard Spearman’s Correlation

The first protocol, labeled ρ -std, first computes the standard Spearman’s ρ correlation between \mathcal{R}_i^g and \mathcal{R}_i^s . The system list \mathcal{R}_i^s is pruned so that it contains only those items that also occur in \mathcal{R}_i^g . The two lists are then correlated to obtain the score ρ_i for cue w_i^c .

Following that, the correlation scores are averaged. First, we apply the Fisher z -transformation (Fisher, 1915) and then average over the transformed scores:

$$z_i = \frac{1}{2} \ln \left(\frac{1 + \rho_i}{1 - \rho_i} \right) = \operatorname{arctanh}(\rho_i) \quad (1)$$

$$z_{avg} = \sum_{i=1}^{|\mathcal{W}^c|} z_i \quad (2)$$

The final output score is obtained by applying the inverse z -transformation on z_{avg} :

$$\rho_{avg} = \tanh(z_{avg}) \quad (3)$$

Averaged Weighted Spearman’s Correlation

The previous protocol treats all ranks equally, despite the fact that the system should be rewarded more for getting the strongest responses correct (and penalised when failing to do so). Therefore, we also experiment with weighted rank correlation measures, which weigh the distance between two ranks, and assign more importance to higher ranks (i.e., in our setting, to stronger associates).

Several weighted correlation metrics have been proposed (Blest, 2000; Pinto da Costa and Soares, 2005; Dancelli et al., 2013; Pinto da Costa, 2015). We show results with the weighted Spearman’s correlation (further labelled ρ -w) from Pinto da Costa (2015).⁷ Let us denote $Q_1 = [Q_{1,1}, Q_{1,2}, \dots, Q_{1,n}]$ and $Q_2 = [Q_{2,1}, Q_{2,2}, \dots, Q_{2,n}]$ two vectors of ranks obtained on a sample of size n . The weighted rank correlation ρ between the vectors is computed as:

⁷We also experimented with other weighted variants, but detected similar trends in reported model rankings.

$$1 - \frac{6 \sum_{i=1}^n (Q_{1,i} - Q_{2,i})((n - Q_{1,i} + 1) + (n - Q_{2,i} + 1))}{n^4 + n^3 - n^2 - n} \quad (4)$$

We refer the interested reader to the relevant literature (Pinto da Costa, 2015) for further details, theoretical implications and property proofs related to Eq. (4). ρ_i scores for all cue words \mathcal{W}^c are then obtained using Eq. (4), and the averaged score ρ_{avg} is computed as before, see Eq. (1)-Eq. (3).

While the two metrics are intuitive and capture the ability of models to correctly rank (a subset of) associates/responses, note that they have deficiencies. They only evaluate the rankings of words occurring in \mathcal{R}_i^g , which effectively reduces the search space V^r to the small subset $\{w_1, \dots, w_{|\mathcal{R}_i^g|}\} \subset V^r$. This effectively means that the final score simply ignores incorrect responses that are ranked highly by a system but that do not occur in \mathcal{R}_i^g . It also does not take into account the actual strength of association.

3.2 IR-Inspired Evaluation

Intuition Another set of evaluation metrics is inspired by the resemblance of the USF data structure to the typical output of ad-hoc IR systems (Manning et al., 2008; Pound et al., 2010). That is, each cue word w^c can be thought of as an input *query* issued against some *target concept collection* V^r , where the goal of our *association retrieval system* is to rank items from the target collection according to their *relevance* (i.e., their association strength) to the issued query. The output of the system is the ranked list \mathcal{R}_i^s of length $|V^r|$, with ground truth relevance assessments provided in \mathcal{R}_i^g .

MRR and MAP The first two metrics assume non-weighted or binary relevance: the retrieved response is either relevant to the issued cue (labeled 1) or it is non-relevant (0). We assume that all responses found in the ground truth lists \mathcal{R}_i^g where $fsg_{i,j} > t$ are relevant responses, where t is a threshold.⁸ We label this reduced set of relevant responses $\mathcal{R}\mathcal{R}_i^g$.

The most lenient evaluation metric is Mean Reciprocal Rank (MRR) (Voorhees, 1999; Craswell,

⁸In our experiments, we impose a simple heuristic and take responses as relevant if they were generated by at least 3 different human subjects in the USF experiments. This heuristic reduces the noise in human answers and provides a more coherent set of responses.

2009). The reciprocal rank of a query response is the multiplicative inverse of the rank of the first relevant answer, and the final score is then averaged over all $|\mathcal{W}^c|$ queries/cues. More formally:

$$MRR(\mathcal{W}^c) = \frac{1}{|\mathcal{W}^c|} \sum_{i=1}^{|\mathcal{W}^c|} \frac{1}{rank_i} \quad (5)$$

where $rank_i$ is the rank position of the first relevant response (i.e., the first response found in the set $\mathcal{R}\mathcal{R}_i^g$) for the cue word w_i^c .

Since MRR cannot assess multiple correct answers and their ranking in the retrieved list, an alternative metric is Mean Average Precision (MAP):

$$MAP(\mathcal{W}^c) = \frac{1}{|\mathcal{W}^c|} \sum_{i=1}^{|\mathcal{W}^c|} AP(w_i^c) \quad (6)$$

$$AP(w_i^c) = \frac{\sum_{k=1}^N P_k \cdot irel_k}{|\mathcal{R}\mathcal{R}_i^g|} \quad (7)$$

Here, $AP(w_i^c)$ denotes Average Precision for query/cue w_i^c , $N \leq |V^r|$ denotes the number of responses retrieved by the system. P_k is the precision at cut-off k in the list, and $irel_k$ is an indicator function which 'turns on' only if the response at rank k is the relevant response (i.e., present in $\mathcal{R}\mathcal{R}_i^g$). The average is computed over all relevant responses, and the non-retrieved relevant responses from V^r get a precision score of 0. $N \ll |V^r|$ is typically used (e.g., standard values are $N = 100$ or $N = 1000$) to reduce the execution time of the evaluation procedure, since it is expected that a good retrieval system should obtain a majority of relevant responses in the first N responses.

Compared to measures from Sect. 3.1, MRR and MAP are better estimators of the model's ability to capture word association, as they operate over the entire search space V^r for each cue word. This effectively means that systems get rewarded if they are able to consistently rank relevant responses higher than non-relevant responses. However, these metrics still rely on binary non-weighted relevance judgements, and are therefore unable to reward models that rank highly relevant responses (i.e., strongly associated responses, see Tab. 1) higher than weakly relevant responses.

NDCG@k In other words, the most expressive evaluation metric should be able to distinguish that cue-response pairs such as (*lunch, dinner*) and (*lunch, food*) should be ranked higher than weakly associated pairs such as (*lunch, box*) or (*lunch,*

sandwich). In addition, the metric should still reward models that rank relevant responses higher than non-relevant ones.

An IR metric which takes all these aspects into account is Discounted Cumulative Gain (DCG) (Järvelin and Kekäläinen, 2002). DCG operates with weighted relevance values: in the USF scenario, these are forward association strengths, i.e., scores $fs_{g_{i,j}}$. The main idea behind using DCG is that highly relevant responses appearing lower in a ranked list should be penalised. The penalty is implemented by reducing the weighted relevance value logarithmically proportional to the position of the particular response. We opt for a more recent variant of DCG which puts more emphasis on retrieving relevant responses (Burges et al., 2005). DCG@k, the DCG score accumulated at a particular rank position k is computed as follows:

$$DCG@k = \sum_{i=1}^k \frac{2^{wrel_i} - 1}{\log_2(i + 1)} \quad (8)$$

$wrel_i$ is the graded relevance of the response at rank i given by the ground truth data, i.e., $fs_{g_{i,j}}$ if the cue-response pair occurs in $\mathcal{R}\mathcal{R}_i^g$, or 0 otherwise.

To make results comparable across different queries, a normalised variant of DCG is typically used. First, all relevant responses are sorted by their graded relevance value, producing the maximum possible DCG at each position k . The score of the ideal ranking at rank k is called Ideal DCG (IDCG@k). NDCG@k for a single query is then:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \quad (9)$$

Finally, the mean NDCG@k is produced for the entire collection \mathcal{W}^c by averaging over all single NDCG@k values. In all experiments we rely on a standard choice for k : NDCG@100, while similar trends are observed with NDCG@10.

4 Experimental Setup and Models

LDA-Based Approach First, we evaluate an approach based on latent topic modeling, rooted in the psychology literature (Steyvers et al., 2004; Griffiths et al., 2007; Steyvers and Griffiths, 2007).⁹ The following quantitative model of word association has been proposed (Griffiths et al., 2007):

⁹Griffiths et al. (2007) also experimented with LSA (Landauer and Dumais, 1997) and found that their LDA-based approach consistently outperformed LSA-based approaches.

$$P(w^r|w^c) = \sum_{i=1}^M P(w^r|to_i)P(to_i|w^c) \quad (10)$$

where w^c is a cue word, $w^r \in V^r$ any concept from the search space, and to_i is the i^{th} latent topic from the set of M topics induced from the corpus data (using LDA). We label this model **LDA-assoc**. The probability scores $P(w^r|to_i)$ select words that are highly descriptive for each particular topic. $P(to_i|w^c)$ scores are computed as in prior work, by assuming topic independence and applying Bayes' rule on the LDA output per-topic word distributions $P(\cdot|to_i)$ (Steyvers and Griffiths, 2007; Vulić and Moens, 2013).¹⁰ We train LDA with 1,000 topics using suggested parameters (Griffiths et al., 2007).

Count-Based Models We evaluate the best performing reduced count-based model from (Baroni et al., 2014). We label this model **count-ppmi-500d**.¹¹ For a more detailed description of the model's training data and setup we refer the reader to the original work and supplementary material.

Vector Space Models We also compare the performance of prominent representation models on the WA USF task. We include: (1) unsupervised models that learn from distributional information in text, including Glove (Pennington et al., 2014) with $d = 50$ and $d = 300$ dimensions (**glove-6B-50d** and **glove-6B-300d**), the skip-gram negative-sampling (SGNS) 300-dimensional vectors (Mikolov et al., 2013) with various contexts (*bow* = bag-of-words; *deps* = dependency contexts) as in (Levy and Goldberg, 2014) and (Schwartz et al., 2015) (**sgns-pw-bow-w2**, **sgns-pw-bow-w5**, **sgns-pw-deps**, **sgns-8b-bow-w2**), and the symmetric-pattern based vectors by Schwartz et al. (2015) (**sympat-500d**); (2) Models that rely on linguistic hand-crafted resources or curated knowledge bases. Here, we use vectors fine-tuned to a paraphrase database (**paragram-25d**,

¹⁰The generative model closely resembles the actual process in the human brain (Griffiths et al., 2007) - when we generate responses, we first tend to associate that word with a related semantic/cognitive concept, i.e., a latent topic (the factor $P(to_i|w^c)$), and then, after establishing the concept, we output a list of words that we consider the most prominent/descriptive for that concept (words with high scores in the factor $P(w^r|to_i)$).

¹¹We have also experimented with simple count-based asymmetric association measures proposed by Michelbacher et al. (2007), estimated using the same corpus as the *count-ppmi-500d* model. We do not report the results with these measures, as they show a very poor performance when compared to all other models in our comparison.

paragram-300d, (Wieting et al., 2015)) further refined using linguistic constraints (**paragram+cf-300d**, (Mrkšić et al., 2016)); (3) Multilingual embedding models from Luong et al. (2015) (**biskip-256d**) and Faruqui and Dyer (2014) (**bicca-512d**). More detailed descriptions of all VSM models are available in the listed papers and supplementary material attached to this work.

USF Data Processing and Parameters Only USF pairs where both words are single word expressions were retained, and the rest was discarded. This yields 4,992 single word queries in total. The total number of finally retained USF pairs is $\approx 70,000$. Note that this evaluation set is by an order of magnitude larger than current benchmarking word pair scoring datasets such as MEN (3000 word pairs in total), SimVerb (3500), SimLex (999) and Rare Words (2034), and thus allows for a truly comprehensive evaluation of quantitative WA models. Only responses generated by at least 3 human subjects in each list of responses are taken as relevant in all experiments (see Foot. 7 in Sect. 3.2), all other (*cue, response*) pairs and pairs not present in the USF data are considered non-relevant.¹²

5 Results and Discussion

Exp. I: Making the Evaluation Tractable Computational complexity is not an issue for standard semantic benchmarks such as SimLex-999 or MEN: these data sets require only N_{gt} similarity computations in total, where N_{gt} is the number of word pairs in each benchmark (999 or 3000). However, complexity plays a major role in the USF evaluation: the system has to compute $|\mathcal{W}^c| \cdot |V^r|$ similarity scores, where $|\mathcal{W}^c| \approx 5,000$, and $|V^r|$ is large for large vocabularies (typically covering $> 100K$ words). In addition, each list of $|V^r|$ has to be sorted according to the WA strength: this means that the complexity is $O(|\mathcal{W}^c| \cdot (|V^r| + |V^r| \log |V^r|))$.

Since this is prohibitively expensive, our solution is to restrict the search space V^r only to words (both cues and responses) occurring in USF: $|V^r| = 10,070$.¹³ Besides the gains in evaluation efficiency, when using the USF vocabulary all models operate over exactly the same search space:

¹²For efficiency reasons with IR metrics, we evaluate results only over the top $N = 1000$ retrieved responses for each cue.

¹³Prior work shows that the USF data represents a good range of distinct semantic phenomena (Hill et al., 2015), which suggests that the USF vocabulary represents a balanced sample of the English vocabulary.

Model	$V^r = 100K$			$V^r = USF$		
	MRR	MAP	NDCG	MRR	MAP	NDCG
<i>glove-6B-50d [4988]</i>	0.233 (4)	0.072 (3)	0.190 (3)	0.318 (5)	0.105 (5)	0.249 (5)
<i>glove-6B-300d [4988]</i>	0.303 (1)	0.112 (1)	0.280 (1)	0.473 (1)	0.183 (1)	0.380 (1)
<i>sgns-pw-bow-w2 [4970]</i>	0.177 (6)	0.047 (7)	0.129 (6)	0.315 (6)	0.098 (6)	0.226 (6)
<i>sgns-pw-bow-w5 [4970]</i>	0.235 (3)	0.066 (5)	0.176 (5)	0.372 (3)	0.122 (4)	0.278 (4)
<i>sgns-pw-deps [4953]</i>	0.164 (8)	0.041 (8)	0.107 (8)	0.281 (8)	0.081 (8)	0.187 (8)
<i>sgns-8b-bow-w2 [4982]</i>	0.239 (2)	0.078 (2)	0.218 (2)	0.452 (2)	0.169 (2)	0.358 (2)
<i>paragram-25d [4902]</i>	0.174 (7)	0.048 (6)	0.121 (7)	0.309 (7)	0.092 (7)	0.198 (7)
<i>paragram+cf-300d [4971]</i>	0.221 (5)	0.067 (4)	0.179 (4)	0.371 (4)	0.130 (3)	0.284 (3)

Table 2: The effects of reducing the search space V^r to speed up the evaluation process. The numbers in parentheses are relative rankings of each model (1-8) according to the particular evaluation metric. The numbers in square brackets report the coverage of each model (the total number of USF queries is 4992).

Model	ρ -std	ρ -w	MRR	MAP	NDCG
LDA-assoc	0.230	0.221	0.153	0.048	0.128
count-ppmi-500d	0.255	0.249	0.294	0.094	0.226
<i>glove-6B-50d</i>	0.280	0.277	0.318	0.105	0.249
<i>glove-6B-300d</i>	0.337	0.339	0.473	0.183	0.380
<i>sgns-pw-bow-w2</i>	0.263	0.259	0.315	0.098	0.226
<i>sgns-pw-bow-w5</i>	0.283	0.280	0.372	0.122	0.278
<i>sgns-pw-deps</i>	0.240	0.234	0.281	0.081	0.187
<i>sgns-8b-bow-w2</i>	0.322	0.324	0.452	0.169	0.358
<i>sympat-500d</i>	0.194	0.189	0.221	0.069	0.180
<i>paragram-25d</i>	0.222	0.217	0.309	0.092	0.198
<i>paragram-300d</i>	0.302	0.298	0.388	0.138	0.300
<i>paragram+cf-300d</i>	0.265	0.268	0.372	0.067	0.179
<i>biskip-256d</i>	0.255	0.253	0.283	0.091	0.212
<i>bicca-512d</i>	0.311	0.310	0.371	0.132	0.303

Table 3: Results on the USF WA task using different evaluation metrics proposed in Sect. 3. $V^r = USF$ for all models. The best results per column are in bold, second best in italic.

therefore, their results are directly comparable as the data coverage bias should be largely mitigated.

To fully support this choice, we perform a simple experiment using a subset of models from Sect. 4. In the first evaluation, V^r contains the most frequent 100K words for all models, where frequency was computed on their respective training data. In the second evaluation, V^r contains only the USF vocabulary words. The results with IR-style metrics are shown in Tab. 2, and similar trends are observed with Spearman’s ρ correlations.

The results support several conclusions. (i) Coverage over cue words is very high for all models (the model with the lowest coverage from Tab. 2 has a coverage of 98.2%). This, along with the same search space (the USF vocabulary) indicates a fair comparison of different models. (ii) Different IR metrics produce consistent model rankings, with a slight variation in the middle of the rankings. Interestingly, the best scoring model is Glove, a model which uses document-level co-occurrence, which steers it towards learning topical similarity. On the

other hand, the worst performing model relies on dependency-based contexts which better capture functional similarity (Levy and Goldberg, 2014) and outperform other context choices in word similarity tasks on SimLex and SimVerb (Melamud et al., 2016; Gerz et al., 2016). (iii) Most importantly, the reduction of V^r again yields consistent rankings with all metrics, which are also fairly consistent with the rankings obtained in the ten times larger 100K search space. Therefore, in all further experiments we use the USF vocabulary as our search space.

Exp. II: Results on USF WA Next, we evaluate all models from Sect. 3 on the WA task. The results with different metrics are summarised in Tab. 3. The results suggest that all proposed evaluation metrics indeed reflect the ability of different models to capture WA. We observe strong correlations of the models’ rankings with all five metrics (Tab. 4). ρ -w is a slightly more conservative metric than ρ -std on average, but it does not affect model rankings at all (see also Tab. 4).

Further, the LDA-based WA model (Griffiths et al., 2007) is largely outperformed by VSM-based approaches. As expected, similar VSMS with more dimensions are more expressive and score higher (e.g., note the scores with *glove* and *paragram* models). Additionally, models trained on larger corpora are also able to improve the overall results (e.g., note the scores with *sgns* trained on the Polyglot Wikipedia (PW, 2B tokens) vs. the 8B *word2vec* corpus). The *paragram* models specialised for similarity tasks are unable to match unsupervised VSMS that train on running text (e.g., *paragram+cf-300d* obtains a SimLex score of 0.74 compared to 0.46 with *sgns-8b-bow-w2*).

Two models using bilingual training (*biskip-256d* and *bicca-512d*) seem unable to match the

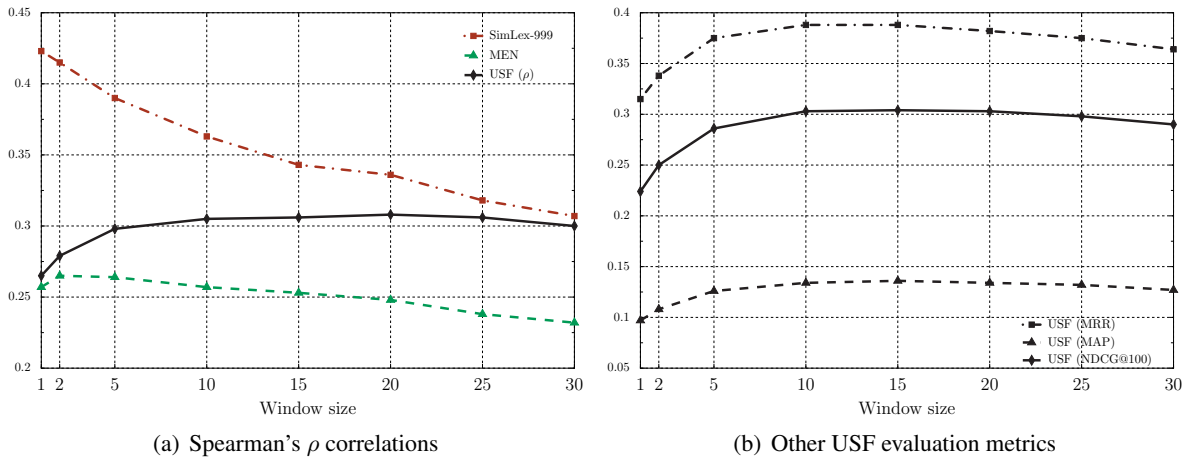


Figure 1: Influence of the window size on the ability of vector space models to capture Similarity (evaluated on SimLex-999), Relatedness (MEN), and Association (USF) (a) Spearman's ρ -std correlations on all three data sets; (b) Behaviour of other evaluation metrics used in the USF evaluation. All tested models are SGNS, $d = 300$, and the only varied hyper-parameter is the window size.

	Association (WA)					Similarity		Relatedness	
	MAP	MRR	NDCG	ρ -std	ρ -w	SimLex	SimVerb	MEN	RareWords
MAP	1.0	0.966	0.986	0.958	0.958	0.088	0.169	0.729	0.645
MRR	0.972	1.0	0.933	0.921	0.921	0.076	0.129	0.626	0.701
NDCG	0.986	0.944	1.0	0.975	0.975	-0.012	0.080	0.722	0.544
ρ -std	0.951	0.923	0.972	1.0	1.0	-0.184	-0.088	0.639	0.425
ρ -w	0.951	0.923	0.972	1.0	1.0	-0.184	-0.088	0.639	0.425
SimLex	0.063	0.098	-0.042	-0.203	-0.203	1.0	0.975	0.370	0.666
SimVerb	0.140	0.098	0.049	-0.111	-0.111	0.972	1.0	0.482	0.667
MEN	0.741	0.657	0.741	0.671	0.671	0.342	0.448	1.0	0.591
RareWords	0.643	0.699	0.538	0.433	0.433	0.622	0.608	0.580	1.0

Table 4: Spearman's ρ correlations between different evaluation protocols for vector space models divided into (a) Association, (b) Similarity, and (c) Relatedness. The correlation scores are based on the rankings of all the evaluated models (see Sect. 4.1) in each experiment. The lower-left part of the table (below the main diagonal, in lighter gray) reports standard Spearman's ρ -std correlations between different model rankings, while ρ -w is reported in the upper-right part (in darker gray). We report model rankings based on the 5 different metrics introduced for the WA USF evaluation. Model rankings for Similarity and Relatedness experiments are according to the ρ -std correlation on the respective ground truth data sets.

best performing monolingual models: however, we plan to further analyse the influence of bilingual information in the WA task in future work.

Finally, a comparison of *sgns-pw*-* models (where the only varied parameter is the context used in training) reveals that (i) larger windows improve WA scores (we test this phenomenon further in Exp. III), (ii) *sgns-pw-deps*, which captures functional similarity through dependency-based contexts, yields lower WA scores, while it improves on SimLex-999 compared to the other two models. This insight leads us to further investigate this phenomenon in Exp. IV.

Exp. III: Window Size In the next experiment, we analysed the effect of the window size on

models' ability to capture similarity, relatedness, and association. We train the *sgns-pw-bow* model ($d = 300$) with varying window sizes in the interval $[1, 30]$. The results on similarity (SimLex-999), relatedness (MEN), and WA benchmarks (USF) are presented in Fig. 1(a)-1(b). It is clear that using larger windows deteriorates the performance on SimLex-999 as the focus of the model is shifted from functional to topical similarity. This shift has been detected in prior work on vector space models (Kiela and Clark, 2014). However, we also observe a similar trend with MEN scores, although an opposite effect was expected, which questions the ability of MEN to accurately evaluate relatedness. The opposite effect is, however, visible with the WA evaluation, where it is evident that larger win-

dows (leading to topical similarity) lead to better WA estimates. This also provides the first hint that WA and semantic similarity capture two completely distinct semantic phenomena.

Exp. IV: WA vs. Similarity vs. Relatedness We delve deeper into this conjecture by computing correlations between model rankings on the WA task and two prominent similarity and relatedness data sets. The results from Tab. 4 indicate the following. First, semantic relatedness and similarity are correlated although they clearly refer to two distinct semantic phenomena as emphasised in prior work (Hill et al., 2015). The correlations between different metrics proposed for the WA task are very high (e.g., the lowest correlation score among any of the two is $\rho = 0.921$). Second, WA and similarity capture very distinct relations (this is evident from low, even negative ρ correlation scores). Third, WA and relatedness are strongly correlated,¹⁴ but the correlation is not as high as expected, given that the two are often considered equivalent, e.g., (Kiela et al., 2015). Future work should investigate whether the difference originates from inadequate evaluation data and protocols (see Fig. 1(a)-1(b) again), or whether the difference is fundamental.

6 Conclusion and Future Work

We have proposed and released a new end-to-end evaluation framework for the task of free word association (WA). We have also provided new evaluation metrics inspired by research in IR, and guidelines for evaluating semantic representation models on the quantitative WA task.

Besides serving as a gold standard in NLP, the comprehensive WA evaluation resource and accompanying evaluation protocol should enable the development of data-driven automatic systems that can capture the notion of word association, and further analysis on how humans perceive (types of) semantic relatedness and similarity (Spence and Owens, 1990; Maki and Buchanan, 2008; De Deyne et al., 2013). These systems, as discussed in this paper, may additionally facilitate research in cognitive psychology pertaining to human semantic representation and memory.

¹⁴Although it comes as slightly counter-intuitive, research in statistics has shown that transitivity between correlation coefficients does not hold in general (Langford et al., 2001; Castro Sotos et al., 2009). Therefore, the observed behaviour is possible: Relatedness indeed correlates both with Association and with Similarity, while at the same time we do not observe any correlation between Association and Similarity.

In future work, we plan to test the portability of the evaluation protocol and apply it to other repositories of word association data in English (De Deyne et al., 2016), as well as in other languages, using existing WA tables in, e.g., German (Schulte im Walde et al., 2008), Dutch (De Deyne and Storms, 2008; Brysbaert et al., 2014), Italian (Guida and Lenci, 2007), Japanese (Joyce, 2005), or Cantonese (Kwong, 2013).¹⁵

In another line of future work, we will experiment with other “cognitively plausible” evaluation data such as N400 (Kutas and Federmeier, 2011; Ettinger et al., 2016), and will analyse the similarities and differences between WA and other such “cognitive” evaluation protocols, as the one relying on semantic priming (SPP) (Hutchison et al., 2013; Ettinger and Linzen, 2016).

All evaluation scripts and detailed guidelines related to this work are freely available at: github.com/cambridge/tl/wa-eval/

Acknowledgments

This work is supported by ERC Consolidator Grant LEXICAL (no 648909). The authors are grateful to the anonymous reviewers for their helpful comments and suggestions.

References

- Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *CoNLL*, pages 183–192.
- Oded Avraham and Yoav Goldberg. 2016. Improving reliability of word similarity evaluation by redesigning annotation task and performance measure. In *REPEVAL*, pages 106–110.
- Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.
- Miroslav Batchkarov, Thomas Kober, Jeremy Reffin, Julie Weeds, and David Weir. 2016. A critique of word similarity as a method for evaluating distributional semantic models. In *REPEVAL*, pages 7–12.
- David C. Blest. 2000. Theory & methods: Rank correlation - an alternative measure. *Australian & New Zealand Journal of Statistics*, 42(1):101–111.
- ¹⁵See also <https://smallworldofwords.org/> for the project aiming to develop WA tables using crowdsourcing in more languages (e.g., Vietnamese, Spanish, French).

- Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.
- Marc Brysbaert, Michaël Stevens, Simon De Deyne, Wouter Voorspoels, and Gert Storms. 2014. Norms of age of acquisition and concreteness for 30,000 Dutch words. *Acta psychologica*, 150:80–84.
- Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to rank using gradient descent. In *ICML*, pages 89–96.
- Ana Elisa Castro Sotos, Stijn Vanhoof, Wim Van Den Noortgate, and Patrick Onghena. 2009. The transitivity misconception of Pearson’s correlation coefficient. *Statistics Education Research Journal*, 8(2):33–55.
- Nick Craswell. 2009. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703.
- Livia Dancelli, Marica Manisera, and Marika Vezzoli. 2013. On two classes of weighted rank correlation measures deriving from the Spearman’s ρ . *Statistical Models for Data Analysis*, pages 107–114.
- Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *ACL*, pages 297–304.
- Simon De Deyne and Gert Storms. 2008. Word associations: Norms for 1,424 Dutch words in a continuous task. *Behavior Research Methods*, 40(1):198–205.
- Simon De Deyne, Daniel J. Navarro, and Gert Storms. 2013. Better explanations of lexical and semantic cognition using networks derived from continued rather than single-word associations. *Behavior Research Methods*, 45(2):480–498.
- Simon De Deyne, Amy Perfors, and Daniel J. Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. In *COLING*, pages 1861–1870.
- James Deese. 1966. *The Structure of Associations in Language and Thought*.
- Simon Dennis and Michael S. Humphreys. 2001. A context noise model of episodic word recognition. *Psychological Review*, 108(2):452–478.
- Allyson Ettinger and Tal Linzen. 2016. Evaluating vector space models using human semantic priming results. In *REPEVAL*, pages 72–77.
- Allyson Ettinger, Naomi H. Feldman, Philip Resnik, and Colin Phillips. 2016. Modeling N400 amplitude using vector space models of word representation. In *Proceedings of the Annual Meeting of the Cognitive Science Society*.
- Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471.
- Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL-HLT*, pages 1606–1615.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.
- Ronald A. Fisher. 1915. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 10(4):507–521.
- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL-HLT*, pages 758–764.
- Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *EMNLP*.
- Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.
- Annamaria Guida and Alessandro Lenci. 2007. Semantic properties of word associations to Italian verbs. *Italian Journal of Linguistics*, 19(2):293–326.
- Amaç Herdağdelen, Katrin Erk, and Marco Baroni. 2009. Measuring semantic relatedness with vector space models and random walks. In *Proceedings of the 2009 Workshop on Graph-based Methods for NLP*, pages 50–53.
- Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can’t see what I mean. In *EMNLP*, pages 255–265.
- Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.
- Keith A. Hutchison, David A. Balota, James H. Neely, Michael J. Cortese, Emily R. Cohen-Shikora, Chi-Shing Tse, Melvin J. Yap, Jesse J. Bengson, Dale Niemeyer, and Erin Buchanan. 2013. The semantic priming project. *Behavior Research Methods*, 45(4):1099–1114.
- Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

- Michael N. Jones, Walter Kintsch, and Douglas J.K. Mewhort. 2006. High-dimensional semantic space accounts of priming. *Journal of Memory and Language*, 55(4):534–552.
- Terry Joyce. 2005. Constructing a large-scale database of Japanese word associations. *Corpus Studies on Japanese Kanji (Glottometrics 10)*, pages 82–98.
- Douwe Kiela and Stephen Clark. 2014. A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality*, pages 21–30.
- Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *ACL*, pages 835–841.
- Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *EMNLP*, pages 2044–2048.
- Marta Kutas and Kara D. Federmeier. 2011. Thirty years and counting: Finding meaning in the N400 component of the event related brain potential (ERP). *Annual Review of Psychology*, 62:621–647.
- Oi Yee Kwong. 2013. Exploring the Chinese mental lexicon with word association norms. In *PACLIC*, pages 153–162.
- Thomas K. Landauer and Susan T. Dumais. 1997. Solutions to Plato’s problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.
- Eric Langford, Neil Schwertman, and Margaret Owens. 2001. Is the property of being positively correlated transitive? *The American Statistician*, 55(4):322–325.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL*, pages 302–308.
- Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.
- William S. Maki and Erin Buchanan. 2008. Latent structure in measures of associative, semantic, and thematic knowledge. *Psychonomic Bulletin & Review*, 15(3):598–603.
- Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2017. Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation. *Journal of Memory and Language*, 92:57–78.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Evaluation in information retrieval. *Introduction to Information Retrieval*, pages 151–175.
- Scott McDonald and Chris Brew. 2004. A distributional model of semantic context effects in lexical processing. In *ACL*, pages 17–24.
- Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *NAACL-HLT*, pages 1030–1040.
- David E. Meyer and Roger W. Schvaneveldt. 1971. Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, 90(2):227–234.
- Lukas Michelbacher, Stefan Evert, and Hinrich Schütze. 2007. Asymmetric association measures. In *RANLP*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.
- Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*, pages 142–148.
- Douglas L. Nelson, Vanessa M. McKinney, Nancy R. Gee, and Gerson A. Janczura. 1998. Interpreting the influence of implicitly activated memories on recall and recognition. *Psychological review*, 105(2):299–324.
- Douglas L. Nelson, Cathy L. McEvoy, and Simon Dennis. 2000. What is free association and what does it measure? *Memory and Cognition*, 28(6):887–899.
- Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.
- Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*, pages 425–430.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

- Joaquim Pinto da Costa and Carlos Soares. 2005. A weighted rank measure of correlation. *Australian & New Zealand Journal of Statistics*, 47(4):515–529.
- Joaquim Pinto da Costa. 2015. *Rankings and Preferences: New Results in Weighted Correlation and Weighted Principal Component Analysis with Applications*.
- Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc object retrieval in the Web of data. In *WWW*, pages 771–780.
- Henry L. Roediger, Jason M. Watson, Kathleen B. McDermott, and David A. Gallo. 2001. Factors that determine false recall: A multiple regression analysis. *Psychonomic Bulletin & Review*, 8(3):385–407.
- Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP*.
- Sabine Schulte im Walde, Alissa Melinger, Michael Roth, and Andrea Weber. 2008. An empirical characterisation of response types in German association norms. *Research on Language and Computation*, 6(2):205–238.
- Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *CoNLL*, pages 258–267.
- Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *EMNLP*, pages 1423–1433.
- Anders Søgaard. 2016. Evaluating word embeddings with fMRI and eye-trackings. In *REPEVAL*, pages 116–121.
- Donald P. Spence and Kimberly C. Owens. 1990. Lexical co-occurrence and association strength. *Journal of Psycholinguistic Research*, 19(5):317–330.
- Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 427(7):424–440.
- Mark Steyvers and Kenneth J. Malmberg. 2003. The effect of normative context variability on recognition memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(5):760–766.
- Mark Steyvers, Richard M. Shiffrin, and Douglas L. Nelson. 2004. Word association spaces for predicting semantic similarity effects in episodic memory. *Experimental Cognitive Psychology and its Applications*, pages 237–249.
- Robert E. Till, Ernest F. Mross, and Walter Kintsch. 1988. Time course of priming for associate and inference words in a discourse context. *Memory & Cognition*, 16(4):283–298.
- Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *EMNLP*, pages 2049–2054.
- Amos Tversky. 1977. Features of similarity. *Psychological Review*, 84(4):327.
- Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In *TREC*, pages 77–82.
- Ivan Vulić and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *NAACL-HLT*, pages 106–116.
- Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2016. HyperLex: A large-scale evaluation of graded lexical entailment. *CoRR*, abs/1608.02117.
- John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*, 3:345–358.

Supplementary Material

Vector Space Models

We evaluate a suite of pre-trained vector space models readily accessible online. We note that these models typically use different training data and other additional resources, and have a varying coverage of the English lexicon, but the evaluation score still reveals their ability to effectively capture word association. As mentioned in the paper, we have aimed at making the comparison fair by evaluating all models using the USF vocabulary as the search space for each model in our comparison.

(0) We evaluate a traditional count-based representation model which uses positive PMI weighting and SVD dimensionality reduction. This is the best performing reduced count-based model from (Baroni et al., 2014). The model was trained on concatenated ukWaC, the English Wikipedia and the British National Corpus with the window size 2, and dimensionality after SVD is set to $d = 500$. Vectors were obtained online.¹⁶ We label this model **count-ppmi-500d**.

(1) Two sets of Glove vectors (Pennington et al., 2014) were used ($d = 50$ and $d = 30$) trained on the 6B corpus of concatenated Wikipedia and GigaWord:¹⁷ **glove-6B-50d** and **glove-6B-300d**.

(2) Pre-trained vectors obtained using skip-gram with negative sampling (SGNS) (Mikolov et al., 2013). We use SGNS vectors from (Levy and Goldberg, 2014): **sgns-pw-bow-w2** and **sgns-pw-bow-w5** denote vectors trained with bag-of-words (BOW) contexts on the Polyglot Wikipedia (PW) (Al-Rfou et al., 2013) with window sizes 2 and 5, respectively; **sgns-pw-deps** denotes vectors trained with dependency-based contexts. All vectors are 300-dimensional.¹⁸ For more details including the preprocessing procedure and the specification of the used dependency parser, we refer the reader to the original work. We evaluate another SGNS-BOW model trained on a large 8B corpus with the window size 2 and $d = 500$ to measure the potential gains stemming from the use of larger training

corpora.¹⁹ This model was used as a baseline in (Schwartz et al., 2015): **sgns-8b-bow-w2**.

(3) A template-based approach to vector space modeling introduced by Schwartz et al. (2015). Vectors are trained based on co-occurrence of words in symmetric patterns (Davidov and Rapoport, 2006). We use pre-trained dense vectors ($d = 500$) trained on the 8B corpus available online:²⁰ **sympat-500d**.

(4) Models that use additional linguistic repositories to build semantically specialised improved word vectors. Wieting et al. (2015) use the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) to learn word vectors which emphasise paraphrasability. They do this by fine-tuning, also known as retro-fitting (Faruqui et al., 2015), SGNS vectors using an objective function designed to incorporate the PPDB semantic similarity constraints. We test two variants of the Paragram model ($d = 25$ and $d = 300$) available online:²¹ **paragram-25d** and **paragram-300d**.

Another variant of the fine-tuning procedure called counter-fitting (CF) was recently proposed by Mrkšić et al. (2016). The model further improves the Paragram vectors by injecting antonymy constraints from PPDB v2.0 (Pavlick et al., 2015) into the final vector space. $d = 300$. We label this model **paragram+cf-300d**.²²

(5) Two multilingual pre-trained embedding models, aiming to test whether multilingual supervision can help in capturing word association the same way it helps semantic similarity tasks. We use pre-trained vectors of (Luong et al., 2015) (**biskip-256d**) which rely on word-aligned parallel data,²³ and CCA-based vectors of Faruqui and Dyer (2014) (**bicca-512d**) which require readily available translation lexicons.²⁴ As bilingual representations are not the main focus of this work, for further training details, we refer the reader to the literature.

¹⁶<http://clic.cimec.unitn.it/composes/semantic-vectors.html>

¹⁷<http://nlp.stanford.edu/projects/glove/>

¹⁸<https://levyomer.wordpress.com/publications/>

¹⁹code.google.com/p/word2vec/source/browse/trunk/demo-train-big-model-v1.sh

²⁰http://homes.cs.washington.edu/~roysch/papers/sp_embeddings/sp_embeddings.html

²¹<http://ttic.uchicago.edu/~wieting/>

²²<https://github.com/nmrksic/counter-fitting>

²³<http://stanford.edu/~lmthang/bivec/>

²⁴<http://www.manalafaruqui.com/>

Computational Argumentation Quality Assessment in Natural Language

Henning Wachsmuth Bauhaus-Universität Weimar Weimar, Germany
henning.wachsmuth@uni-weimar.de

Nona Naderi University of Toronto Toronto, Canada
nona@cs.toronto.edu

Yufang Hou IBM Research Dublin, Ireland
yhou@ie.ibm.com

Yonatan Bilu IBM Research Haifa, Israel
yonatanb@il.ibm.com

Vinodkumar Prabhakaran Stanford University Stanford, CA, USA
vinod@cs.stanford.edu

Tim Alberdingk Thijm, Graeme Hirst University of Toronto Toronto, Canada
{thijm, gh}@cs.toronto.edu

Benno Stein Bauhaus-Universität Weimar Weimar, Germany
benno.stein@uni-weimar.de

Abstract

Research on computational argumentation faces the problem of how to automatically assess the quality of an argument or argumentation. While different quality dimensions have been approached in natural language processing, a common understanding of argumentation quality is still missing. This paper presents the first holistic work on computational argumentation quality in natural language. We comprehensively survey the diverse existing theories and approaches to assess logical, rhetorical, and dialectical quality dimensions, and we derive a systematic taxonomy from these. In addition, we provide a corpus with 320 arguments, annotated for all 15 dimensions in the taxonomy. Our results establish a common ground for research on computational argumentation quality assessment.

1 Introduction

What is a good argument? What premises should it be based on? When is argumentation persuasive? When is it reasonable? We subsume such questions under the term *argumentation quality*; they have driven logicians, rhetoricians, linguists, and argumentation theorists since the Ancient Greeks (Aristotle, 2007). Now that the area of computational argumentation is seeing an influx of research activity, the automatic assessment of argumentation quality is coming into the focus, due to its importance for envisioned applications such as writing support (Stab and Gurevych, 2014) and argument search (Wachsmuth et al., 2017), among others.

Existing research covers the mining of argument units (Al-Khatib et al., 2016), specific types of evidence (Rinott et al., 2015), and argumentative relations (Peldszus and Stede, 2015). Other works clas-

sify argumentation schemes (Feng et al., 2014) and frames (Naderi and Hirst, 2015), analyze overall argumentation structures (Wachsmuth et al., 2015), or generate claims (Bilu and Slonim, 2016). Also, theories of argumentation quality exist, and some quality dimensions have been assessed computationally (see Section 2 for details). Until now, however, the assertion of O’Keefe and Jackson (1995) that there is neither a general idea of what constitutes argumentation quality in natural language nor a clear definition of its dimensions still holds.

The reasons for this deficit originate in the varying goals of argumentation: persuading audiences, resolving disputes, achieving agreement, completing inquiries, and recommending actions (Tindale, 2007). As a result, diverse quality dimensions play a role, which relate to the logic of arguments, to the style and rhetorical effect of argumentation, or to its contribution to a discussion. Consider the following argument against the death penalty:¹

Everyone has an inalienable human right to life, even those who commit murder; sentencing a person to death and executing them violates that right.

Although implicit, the conclusion about the death penalty seems sound in terms of (informal) logic, and the argument is clear from a linguistic viewpoint. Some people might not accept the first stated premise, though, especially if emotionally affected by some legal case at hand. Or, they might not be persuaded that the stated argument is the most relevant in the debate on death penalty.

This example reveals three central challenges: (1) Argumentation quality is assessed on different levels of granularity; (2) many quality dimensions are subjective, depending on preconceived opinions; and (3) overall argumentation quality seems hard to measure, as the impact and interaction of the different dimensions remain unclear.

¹Taken from www.bbc.co.uk/ethics/capitalpunishment.

This paper does *not* propose a specific approach to assess quality; rather it defines a common ground by providing a so-far-missing holistic view on argumentation quality assessment in natural language. In particular, we first briefly but comprehensively survey all major theories and computational approaches for argumentation quality. Following Blair (2012), we distinguish three main quality aspects, each associated with several quality dimensions:

- *Logical quality* in terms of the cogency or strength of an argument.
- *Rhetorical quality* in terms of the persuasive effect of an argument or argumentation.
- *Dialectical quality* in terms of the reasonableness of argumentation for resolving issues.

We organize the survey along these aspects, discussing quality at four levels of granularity: (1) *argument unit*, i.e., a segment of text that takes the role of a premise or conclusion; (2) *argument*, i.e., a composition of premises and a conclusion, some of which may be implicit; (3) (*monological*) *argumentation*, i.e., a composition of arguments on a given issue; and (4) (*dialogical*) *debate*, i.e., a series of interacting argumentation on the same issue.

To unify and to consolidate existing research, we then derive a generally applicable taxonomy of argumentation quality from the survey. The taxonomy systematically decomposes quality assessment based on the interactions of 15 widely accepted quality dimensions (including the overall quality). Moreover, we provide a new annotated corpus with 320 arguments for which three experts assessed all 15 dimensions, resulting in over 14,000 annotations. Our analysis indicates how the dimensions interact and which of them are subjective, making the corpus an adequate benchmark for future research.

In summary, the contributions of this paper are:

1. A *comprehensive survey* of research on argumentation quality assessment (Section 2).
2. A *taxonomy* of all major quality dimensions of natural language argumentation, which clarifies their roles and dependencies (Section 3).
3. An *annotated corpus* for computational argumentation quality assessment (Section 4).²

2 Survey of Argumentation Quality

This section briefly surveys all major existing theories and the assessment of natural language argu-

mentation quality. While we order the discussions along the three main quality aspects, we point out overlaps and interrelations where relevant.

2.1 Theories of Argumentation Quality

We focus on the major fields dealing with argumentation quality in natural language: argumentation theory and rhetoric. Table 1 gives an overview of the quality dimensions that we detail below.

Logic Formal argumentation studies the *soundness* of arguments, requiring the truth of an argument's premises and the deductive *validity* of inferring its conclusion. In case of inductive strength, the conclusion becomes probable given the premises. While sound arguments exist in natural language, most are defeasible in nature (Walton, 2006). The desired property of such arguments is *cogency*.

A cogent (or logically good) argument has individually acceptable premises that are relevant to the argument's conclusion and, together, sufficient to draw the conclusion (Johnson and Blair, 2006). Here, (*local*) *acceptability* means that a premise is rationally worthy of being believed by the target audience of the argument. It replaces truth, which is often unclear (Hamblin, 1970). A premise's (*local*) *relevance* refers to the level of support it provides for the conclusion, and (*local*) *sufficiency* captures whether the premises give enough reason to accept the conclusion. In the end, sufficiency thus presupposes relevance (Blair, 2012). While acceptability is more dialectical, overall the three dimensions of cogency are, with slight variations, acknowledged to cover the logical quality of arguments.

Damer (2009) adds that a good argument also depends on the rebuttal it gives to anticipated counterarguments (a dialectical property) as well as on its structural *well-formedness*, i.e., whether it is intrinsically consistent, avoids begging the question, and uses a valid inference rule. These dimensions adopt ideas from the argument model of Toulmin (1958), including rebuttals and warrants, and from the argumentation schemes of Walton et al. (2008), whose critical questions are meant to evaluate inference rules. While not focusing on quality, critical questions particularly help identify fallacies.

Introduced by Aristotle as invalid arguments, fallacies have been brought back to attention by Hamblin (1970). In general, a fallacy has some sort of error in reasoning (Tindale, 2007). Fallacies range from resorting to inapplicable evidence types or irrelevant premises to rhetoric-related errors, such

²The corpus is freely available at: <http://www.arguana.com>

Aspect	Quality Dimension	Granularity	Sources
Logic	Cogency	Argument	Johnson and Blair (2006), Damer (2009), Govier (2010)
	Local relevance	Argument (unit)	Johnson and Blair (2006), Damer (2009), Govier (2010)
Dialectic	Local sufficiency	Argument	Johnson and Blair (2006), Damer (2009), Govier (2010)
	Well-Formedness	Argument	Walton et al. (2008), Damer (2009)
Dialectic	Global sufficiency	Argument	Toulmin (1958), Damer (2009)
	Local acceptability	Argument (unit)	Johnson and Blair (2006), Damer (2009), Govier (2010)
Dialectic	Fallaciousness	Argument (unit)	Hamblin (1970), Tindale (2007), Walton et al. (2008)
	Local relevance	Argument (unit)	Hamblin (1970), Tindale (2007)
Dialectic	Local sufficiency	Argument	Hamblin (1970), Tindale (2007)
	Validity	Argument	Hamblin (1970), Tindale (2007)
Dialectic	Well-Formedness	Argument	Hamblin (1970), Tindale (2007)
	Strength	Argument	Perelman et al. (1969), Tindale (2007), Freeman (2011)
Rhetoric	Effectiveness	Argument(ation)	Perelman et al. (1969), O’Keefe and Jackson (1995)
	Arrangement	Argumentation	Aristotle (2007), Damer (2009)
Rhetoric	Appropriateness of style	Argumentation	Aristotle (2007)
	Clarity of style	Argumentation	Aristotle (2007), Tindale (2007), Govier (2010)
Rhetoric	Credibility	Argumentation	Aristotle (2007)
	Emotional appeal	Argumentation	Aristotle (2007), Govier (2010)
Logic	Soundness	Argument	Aristotle (2007)
Dialectic	Convincingness	Argumentation	Perelman et al. (1969)
	Global acceptability	Argument(ation)	Perelman et al. (1969)
Dialectic	Reasonableness	Argumentation, debate	van Eemeren and Grootendorst (2004)
	Global acceptability	Argument(ation)	van Eemeren and Grootendorst (2004)
Dialectic	Global relevance	Argument(ation)	van Eemeren and Grootendorst (2004), Walton (2006)
	Global sufficiency	Argumentation, debate	Cohen (2001)

Table 1: Theoretical treatment of quality dimensions in the referenced sources for the given granularities of natural language argumentation, grouped by the aspect the bold-faced high-level dimensions refer to.

as unjustified appeals to emotion. They represent an alternative assessment of logical quality. Following Damer (2009), a fallacy can always be seen as a violation of one or more dimensions of good arguments. *Fallaciousness* negatively affects an argument’s *strength* (Tindale, 2007).

Argument strength is often referred to, but its meaning remains unclear: “Is a strong argument an effective argument which gains the adherence of the audience, or is it a valid argument, which ought to gain it?” (Perelman et al., 1969). Tindale (2007) sees validity as a possible but not mandatory part of reasoning strength. Freeman (2011) speaks of the strength of support, matching the idea of inductive strength. Blair (2012) roughly equates strength with cogency, and Hoeken (2001) observes correlations between evidence strength and rhetorical persuasiveness. Such dependencies are expected, as the use of true and valid arguments represents one means of persuasion: *logos* (Aristotle, 2007).

Rhetoric Aristotle’s work on rhetoric is one of the most systematic to this day. He defines rhetoric as the ability to know how to persuade (Aristotle, 2007). Besides *logos*, the three means of persuasion he sees include *ethos*, referring to the arguer’s *credibility*, and *pathos*, the successful *emotional appeal* to the target audience. Govier (2010) outlines how emotions interfere with logic in arguments.

Pathos is not necessarily reprehensible; it just aims for an emotional state adequate for persuasion.

In overall terms, rhetorical quality is reflected by the persuasive *effectiveness*, i.e., the success in persuading a target audience of a conclusion (Blair, 2012). It has been suggested that what arguments are considered as effective is subjective (O’Keefe and Jackson, 1995). Unlike persuasiveness, which relates to the actual arguments, effectiveness covers all aspects of an argumentation, including the use of language (van Eemeren, 2015). In particular, the three means of persuasion are meant to be realized by what is said and how (Aristotle, 2007). Several linguistic quality dimensions are connected to argumentation (examples follow in Section 2.2). While many of them are distinguished by Aristotle, he groups them as the *clarity* and the *appropriateness* of style as well as the proper *arrangement*.

Clarity means the use of correct, unambiguous language that avoids unnecessary complexity and deviation from the discussed issue (Aristotle, 2007). Besides ambiguity, vagueness is a major problem impairing clarity (Govier, 2010) and can be a cause of fallacies (Tindale, 2007). So, clarity is a prerequisite of *logos*. Also, it affects credibility, since it indicates the arguer’s skills. An appropriate style in terms of the choice of words supports credibility and emotions. It is tailored to the issue and

audience (Aristotle, 2007). Arrangement, finally, addresses the structure of argumentation regarding the presentation of the issue, pros, cons, and conclusions. Damer (2009) outlines that a proper arrangement is governed by the dimensions of a good argument. To be effective, well-arranged argumentation matches the expectations of the target audience and is, thus, related to dialectic (Blair, 2012).

Dialectic The dialectical view of argumentation targets the resolution of differences of opinions on the merit (van Eemeren and Grootendorst, 2004). Quality is assessed for well-arranged discussions that seek agreement. In contrast to the subjective nature of effectiveness, people are good in such an assessment (Mercier and Sperber, 2011). In their pragma-dialectical theory, van Eemeren and Grootendorst (2004) develop rules for obtaining *reasonableness* in critical discussions. Reasonableness emerges from two complementary dimensions, intersubjective (*global*) *acceptability* and problem-solving validity, but effectiveness still remains the underlying goal (van Eemeren, 2015). For argumentation, global acceptability is given when the stated arguments and the way they are stated are acceptable to the whole target audience. Problem-solving validity matches the (*global*) *relevance* of argumentation that contributes to resolution, helping arrive at an ultimate conclusion (Walton, 2006).

Global relevance implicitly excludes fallacious moves, so reasonable arguments are cogent (van Eemeren, 2015). Van Eemeren sees reasonableness as a precondition for *convincingness*, the rational version of persuasiveness. Following Perelman et al. (1969), persuasive argumentation aims at a particular audience, whereas convincing argumentation aims at the universal audience, i.e., all reasonable beings. This fits the notion that dialectic examines general rather than specific issues (Aristotle, 2007).

Convincingness needs (*global*) *sufficiency*, i.e., all objections to an argumentation are countered. The dilemma here is that the number of objections could be infinite, but without global sufficiency the required support seems arbitrary (Blair, 2012). A solution is the relaxed view of Damer (2009) that only those counter-arguments that can be anticipated are to be rebutted. For debates, Cohen (2001) speaks of dialectical satisfactoriness, i.e., whether all questions and objections have been sufficiently answered. In case a reasonable debate ends up in either form of global sufficiency, this implies that the discussed difference of opinion is resolved.

Other Although closely related, critical thinking (Freeley and Steinberg, 2009) and persuasion research (Zhao et al., 2011) are covered only implicitly here; their views on quality largely match with argumentation theory. We have not discussed deliberation, as it is not concerned with the quality of argumentation primarily but rather with communicative dimensions of group decision-making, e.g., participation and respect (Steenbergen et al., 2003). Also, we have restricted our view to the logic found in natural language. For formal and probabilistic logic, dimensions such as degree of justification (Pollock, 2001), argument strength (Pfeifer, 2013), and premise relevance (Ransom et al., 2015) have been analyzed. As we see below, such logic influenced some practical assessment approaches.

2.2 Approaches to Quality Assessment

As for the theories, we survey the automatic quality assessment for natural language argumentation. All discussed approaches are listed in Table 2.

Logic Braunstain et al. (2016) deal with logical argument quality in community question answering: Combining relevance-oriented retrieval models and argument-oriented features, they rank sentence-level argument units according to the *level of support* they provide for an answer. Unlike classical essay scoring, Rahimi et al. (2014) score an essay's *evidence*, a quality dimension of argumentation: it captures how sufficiently the given details support the essay's thesis. On the dataset of Correnti et al. (2013) with 1569 student essays and scores from 1 to 4, they find that the concentration and specificity of words related to the essay prompt (i.e., the statement defining the discussed issue) impacts scoring accuracy. Similarly, Stab and Gurevych (2017) introduce an essay corpus with 1029 argument-level annotations of *sufficiency*, following the definition of Johnson and Blair (2006). Their experiments suggest that convolutional neural networks outperform feature-based sufficiency classification.

Rhetoric Persing et al. (2010) tackle the proper arrangement of an essay, namely, its *organization* in terms of the logical development of an argument. The authors rely on manual 7-point score annotations for 1003 essays from the ICLE corpus (Granger et al., 2009). In their experiments, sequences of paragraph discourse functions (e.g., introduction or rebuttal) turn out to be most effective. Organization is also analyzed by Rahimi et al. (2015) on the same dataset used for the evidence

Aspect	Quality Dimension	Granularity	Text Genres	Sources
Logic	Evidence	Argumentation	Student essays	Rahimi et al. (2014)
	Level of support	Argument unit	Wikipedia articles	Braunstein et al. (2016)
	Sufficiency	Argument	Student essays	Stab and Gurevych (2017)
Rhetoric	Argument strength	Argumentation	Student essays	Persing and Ng (2015)
	Evaluability	Argumentation	Law comments	Park et al. (2015)
	Global coherence	Argumentation	Student essays	Feng et al. (2014)
	Organization	Argumentation	Student essays	Persing et al. (2010), Rahimi et al. (2015)
	Persuasiveness	Argument	Forum discussions	Tan et al. (2016), Wei et al. (2016)
	Prompt adherence	Argumentation	Student essays	Persing and Ng (2014)
	Thesis clarity	Argumentation	Student essays	Persing and Ng (2013)
	Winning side	Debate	Oxford-style debates	Zhang et al. (2016)
Dialectic	Acceptability	Argument	Debate portal arguments	Cabrio and Villata (2012)
	Convincingness	Argument	Debate portal arguments	Habernal and Gurevych (2016)
	Prominence	Argument	Forum discussions	Boltužić and Šnajder (2015)
	Relevance	Argument	Diverse genres	Wachsmuth et al. (2017)

Table 2: Practical assessment of quality dimensions in the referenced sources for the given granularities and text genres of natural language argumentation, grouped by the aspect the quality dimensions refer to.

approach above. Their results indicate a correlation between organization and local coherence. Feng et al. (2014) parse discourse structure to assess *global coherence*, i.e., the continuity of meaning in a text. Lacking ground-truth coherence labels, they evaluate their approach on sentence ordering and organization scoring instead. Coherence affects the clarity of style, as do the *thesis clarity* and *prompt adherence* of essays. Persing and Ng (2013) find the former to suffer from misspellings, while Persing and Ng (2014) use prompt-related keywords and topic models to capture the latter (both for 830 ICLE essays like those mentioned above). For comments in lawmaking, Park et al. (2015) develop an argumentation model that prescribes what information users should give to achieve *evaluability* (e.g., testimony evidence or references to resources).

Not only linguistic quality, but also effectiveness is assessed in recent work: Persing and Ng (2015) score the *argument strength* of essays, which they define rhetorically in terms of how many readers would be persuaded. Although potentially subjective, their manual 7-point score annotations of 1000 ICLE essays differ by at most 1 in 67% of the studied cases. Their best features are heuristic argument unit labels and part-of-speech n-grams. Recently, Wachsmuth et al. (2016) demonstrated that the output of argument mining helps in such argumentation-related essay scoring, obtaining better results for argument strength and organization. Tan et al. (2016) analyze which arguments achieve *persuasiveness* in “change my view” forum discussions, showing that multiple interactions with the view-holder are beneficial as well as an appropriate style and a high number of participants. On similar

data, Wei et al. (2016) find that also an author’s reputation impacts persuasiveness. Zhang et al. (2016) discover for Oxford-style debates that attacking the opponents’ arguments tends to be more effective than relying on one’s own arguments. These results indicate the relation of rhetoric and dialectic.

Dialectic Dialectical quality has been addressed by Cabrio and Villata (2012). The authors use textual entailment to find ground-truth debate portal arguments that attack others. Based on the formal argumentation framework of Dung (1995), they then assess global argument *acceptability*. Habernal and Gurevych (2016) compare arguments in terms of *convincingness*. However, the subjective nature of their crowdsourced labels actually reflects rhetorical effectiveness. Boltužić and Šnajder (2015) present first steps towards argument *prominence*. Prominence may be a product of popularity, though, making its quality nature questionable, as popularity is often not correlated with merit (Govier, 2010). In contrast, Wachsmuth et al. (2017) adapt the famous PageRank algorithm to objectively derive the *relevance* of an argument at web scale from what other arguments refer to the argument’s premises. On a large ground-truth argument graph, their approach beats several baselines for the benchmark argument rankings that they provide.

Other Again, we have left out deliberative quality (Gold et al., 2015). Also, we omit approaches that classify argumentation schemes (Feng and Hirst, 2011), evidence types (Rinott et al., 2015), ethos-related statements (Duthie et al., 2016), and myside bias (Stab and Gurevych, 2016); their output may help assess quality assessment, but they do not actually assess it. The same holds for argument mining,

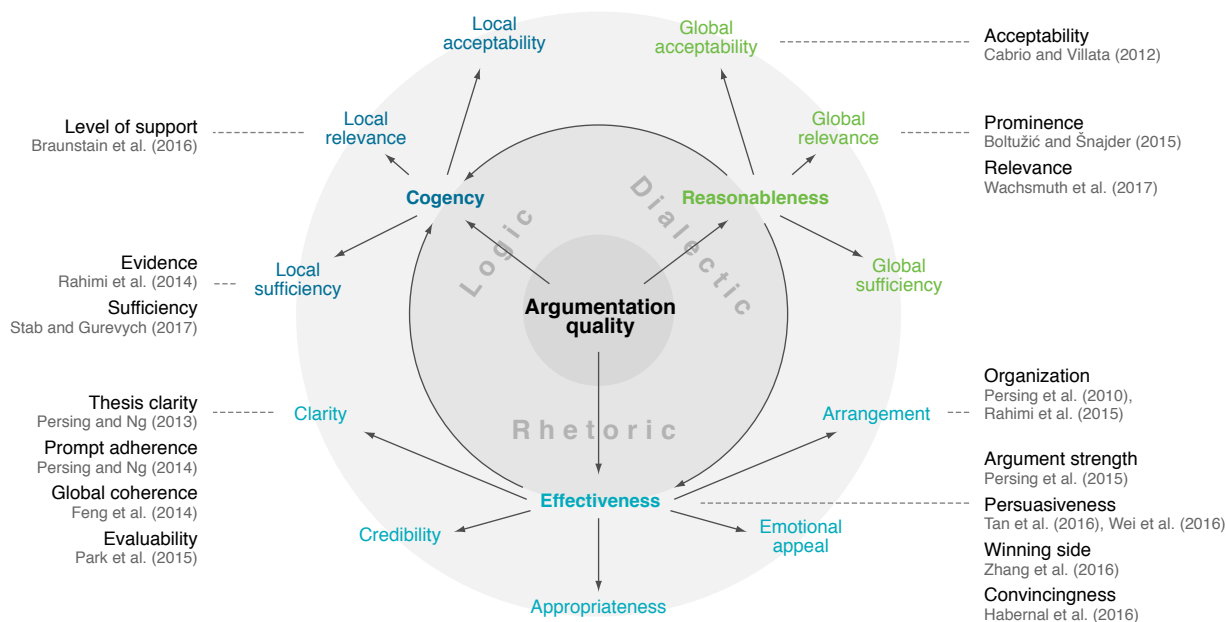


Figure 1: The proposed taxonomy of argumentation quality as well as the mapping of existing assessment approaches to the covered quality dimensions. Arrows show main dependencies between the dimensions.

even if said to aim for argument quality (Swanson et al., 2015). Much work exists for general text quality, most notably in the context of readability (Pitler and Nenkova, 2008) and classical essay scoring. Some scoring approaches derive features from discourse (Burstein et al., 1998), arguments (Ong et al., 2014; Beigman Klebanov et al., 2016; Ghosh et al., 2016), or schemes (Song et al., 2014)—all this may be indicative of quality. However, our focus is approaches that target argumentation quality at heart. Similarly, review helpfulness (Liu et al., 2008) and deception (Ott et al., 2011) are not treated, as arguments only partly play a role there. Also, only few Wikipedia quality flaws relate to arguments, e.g., verifiability (Anderka et al., 2012).

3 A Taxonomy of Argumentation Quality

Given all surveyed quality dimensions, we now propose a unifying taxonomy of argumentation quality. The taxonomy decomposes quality assessment systematically, thus organizing and clarifying the roles of practical approaches. It does not require a particular argumentation model, but it rests on the notion of the granularity levels from Section 1.

3.1 Overview of the Theory-based Taxonomy

Our objective is not to come up with a new theory, but to provide a unified view of existing theories that is suitable for quality assessment. We aim for a common understanding of the dimensions that af-

fect quality, what interdependencies they have, and how they interact. Figure 1 illustrates the taxonomy that we propose for this purpose. The rationale behind its structure and its layout is as follows.

While Section 2 has outlined overlaps and relations between the three aspects of argumentation, we have identified one dominant high-level quality dimension of *argumentation quality* in theory for each aspect: logical *cogency*, rhetorical *effectiveness*, and dialectical *reasonableness*. The latter two benefit from cogency, and reasonableness depends on effectiveness, as discussed. Often, only one of them will be in the focus of attention in practice, or even only a sub-dimension. In particular, each high-level dimension has a set of sub-dimensions agreed upon. The sub-dimensions are shown on the outer ring in Figure 1, roughly positioned according to the aspects they refer to, e.g., *local acceptability* lies next to the other dialectical dimensions. We ordered the sub-dimensions by their interrelations (left implicit for conciseness), e.g., *appropriateness* supports *credibility* and *emotional appeal*.

Slightly deviating from theory, we match Aristotle’s *logos* dimension with cogency, which better fits real-world argumentation. Similarly, we omit those dimensions from Table 1 in the taxonomy that have unclear definitions, such as strength, or that are covered by others, such as well-formedness, which merely refines the acceptability part of cogency (Govier, 2010). Convincingness is left out,

as it is close to effectiveness and as both the feasibility and the need of persuading the universal audience has been questioned (van Eemeren, 2015). Instead, we add *global sufficiency* as part of reasonableness. While global sufficiency may be infeasible, too (Blair, 2012), it forces agreement in critical discussions and, thereby, reasonableness.

3.2 Definitions of the Quality Dimensions

Cogency is seen as an argument property, whereas effectiveness and reasonableness are assessed on the argumentation level usually. For generality, we give informal literature-based definitions of these dimensions and all sub-dimensions here for an author who argues about an issue to a target audience:

Cogency An argument is cogent if it has acceptable premises that are relevant to its conclusion and that are sufficient to draw the conclusion.

- *Local acceptability*: A premise of an argument is acceptable if it is rationally worthy of being believed to be true.
- *Local relevance*: A premise of an argument is relevant if it contributes to the acceptance or rejection of the argument’s conclusion.
- *Local sufficiency*: An argument’s premises are sufficient if, together, they give enough support to make it rational to draw its conclusion.

Effectiveness Argumentation is effective if it persuades the target audience of (or corroborates agreement with) the author’s stance on the issue.

- *Credibility*: Argumentation creates credibility if it conveys arguments and similar in a way that makes the author worthy of credence.
- *Emotional Appeal*: Argumentation makes a successful emotional appeal if it creates emotions in a way that makes the target audience more open to the author’s arguments.
- *Clarity*: Argumentation has a clear style if it uses correct and widely unambiguous language as well as if it avoids unnecessary complexity and deviation from the issue.
- *Appropriateness*: Argumentation has an appropriate style if the used language supports the creation of credibility and emotions as well as if it is proportional to the issue.
- *Arrangement*: Argumentation is arranged properly if it presents the issue, the arguments, and its conclusion in the right order.

Reasonableness Argumentation is reasonable if it contributes to the issue’s resolution in a sufficient way that is acceptable to the target audience.

- *Global acceptability*: Argumentation is acceptable if the target audience accepts both the consideration of the stated arguments for the issue and the way they are stated.
- *Global relevance*: Argumentation is relevant if it contributes to the issue’s resolution, i.e., if it states arguments or other information that help to arrive at an ultimate conclusion.
- *Global sufficiency*: Argumentation is sufficient if it adequately rebuts those counter-arguments to it that can be anticipated.

3.3 Organization of Assessment Approaches

The taxonomy is meant to define a common ground for assessing argumentation quality, including the organization of practical approaches. The left and right side of Figure 1 show where the approaches surveyed in Section 2.2 are positioned in the taxonomy. Some dimensions have been tackled multiple times (e.g., *clarity*), others not at all (e.g., *credibility*). The taxonomy indicates what sub-dimensions will affect the same high-level dimension.

4 The Dagstuhl-15512 ArgQuality Corpus

Finally, we present our new annotated *Dagstuhl-15512 ArgQuality Corpus* for studying argumentation quality based on the developed taxonomy, and we report on a first corpus analysis.³

4.1 Data and Annotation Process

Our corpus is based on the *UKPConvArgRank* dataset (Habernal and Gurevych, 2016), which contains rankings of 25 to 35 textual debate portal arguments for two stances on 16 issues, such as *evolution vs. creation* and *ban plastic water bottles*. All ranks were derived from crowdsourced convincingness labels. For every issue/stance pair, we took the five top-ranked texts and chose five further via stratified sampling. Thereby, we covered both high-quality arguments and different levels of lower quality. Two example texts follow below in Figure 2.

Before annotating the 320 chosen texts, we carried out a full annotation study with seven authors of this paper on 20 argumentative comments from

³The corpus and annotation guidelines are available at <http://www.arguana.com>. The corpus is named after the Dagstuhl Seminar 15512 “Debating Technologies” that initialized the research in this paper: <http://www.dagstuhl.de/15512>

Quality Dimension	(a) Maj. Scores			(b) Agreement			(c) Pearson Correlation Coefficients													
	1	2	3	α	full	maj.	Co	LA	LR	LS	Ef	Cr	Em	Cl	Ap	Ar	Re	GA	GR	GS
Co Cogency	150	131	23	.44	40.1%	91.8%	.64	.61	.84	.81	.46	.27	.41	.32	.55	.78	.64	.71	.70	
LA Local acceptability	84	169	51	.46	27.0%	90.8%	.64	.51	.53	.60	.54	.30	.40	.54	.46	.68	.75	.46	.45	
LR Local relevance	25	155	124	.47	32.6%	92.4%	.61	.51	.56	.56	.39	.27	.46	.35	.50	.62	.58	.68	.45	
LS Local sufficiency	172	119	13	.44	37.2%	92.8%	.84	.53	.56	.73	.39	.25	.37	.23	.51	.67	.51	.68	.74	
Ef Effectiveness	184	111	9	.45	42.1%	94.4%	.81	.60	.56	.73	.48	.31	.35	.34	.54	.75	.58	.66	.71	
Cr Credibility	99	199	6	.37	37.8%	95.7%	.46	.54	.39	.39	.48	.37	.32	.49	.37	.52	.52	.36	.40	
Em Emotional appeal	48	235	21	.26	42.8%	94.4%	.27	.30	.27	.25	.31	.37	.14	.30	.20	.30	.26	.26	.22	
Cl Clarity	42	191	71	.35	29.3%	89.8%	.41	.40	.46	.37	.35	.32	.14	.45	.56	.44	.45	.38	.27	
Ap Appropriateness	43	196	65	.36	17.4%	87.5%	.32	.54	.35	.23	.34	.49	.30	.45	.48	.47	.59	.20	.20	
Ar Arrangement	91	189	24	.39	26.6%	93.4%	.55	.46	.50	.51	.54	.37	.20	.56	.48	.55	.51	.49	.48	
Re Reasonableness	126	159	19	.50	41.4%	95.7%	.78	.68	.62	.67	.75	.52	.30	.44	.47	.55	.78	.65	.61	
GA Global acceptability	88	161	55	.44	31.6%	95.4%	.64	.75	.58	.51	.58	.52	.26	.45	.59	.51	.78	.46	.43	
GR Global relevance	69	167	68	.42	21.7%	90.1%	.71	.46	.68	.68	.66	.36	.26	.38	.20	.49	.65	.46	.61	
GS Global sufficiency	231	72	1	.27	44.7%	98.0%	.70	.45	.45	.74	.71	.40	.22	.27	.20	.48	.61	.43	.61	
Ov Overall quality	152	128	24	.51	44.1%	94.4%	.84	.66	.61	.74	.81	.52	.30	.45	.42	.59	.86	.71	.70	.68

Table 3: Results for the 304 corpus texts classified as argumentative by all annotators: (a) Distribution of majority scores for each dimension (2 used in case of full disagreement). (b) Krippendorff’s α of the most agreeing annotator pair and full/majority agreement of all annotators. (c) Correlation for each dimension pair, averaged over the correlations of all annotators. The highest value in each column is marked bold.

the unshared task dataset of the 3rd Workshop on Argument Mining.⁴ The annotators assessed all 15 quality dimensions in the taxonomy for each comment (including its overall quality). Due to simple initial guidelines based on the definitions from Section 3 and the subjectiveness of the task, the agreement of all seven annotators was low for all dimensions, namely, at most .22 in terms of Krippendorff’s α . The three most agreeing annotators for each dimension achieved much higher α -values between .23 (clarity) and .60 (credibility), though.⁵

The study results were discussed by all annotators, leading to a considerably refined version of the guidelines. We then selected three annotators for the corpus annotation based on their availability. They work at two universities and one company in three countries (two females, one male; two PhDs, one PhD student). For each text in the corpus, all annotators first classified whether it was actually argumentative. If so, they assessed all dimensions using ordinal scores from 1 (low) to 3 (high).⁶ Additionally, “cannot judge” could be chosen.

4.2 Corpus Distribution and Agreement

Table 3(a) lists the majority scores of each dimension for the 304 corpus texts (95%) that are classified as argumentative by all annotators, all covering

⁴Unshared task data found at: <http://github.com/UKPLab>

⁵We use Krippendorff’s α as is suitable for small samples, multiple ratings, and ordinal scales (Krippendorff, 2007).

⁶We chose a 3-point scale to foster clear decisions on the quality; in the annotation study, we used a 4-point scale but observed that the annotators only rarely chose score 1 and 4.

the whole score range. Five dimensions have the median at score 1, the others at 2. Some seem easier to master, such as *local relevance*, which received the highest majority score 124 times. Others rarely got score 3, above all *global sufficiency*. The latter is explained by the fact that only few texts include any rebuttal of counter-arguments.

Only one of the over 14,000 assessments made by the three annotators was “cannot judge” (for *global relevance*), suggesting that our guidelines were comprehensive. Regarding agreement, we see in Table 3(b) that the α -values of all logical and dialectical quality dimensions except for *global sufficiency* lie above 0.4 for the most agreeing annotator pair. As expected, the rhetorical dimensions seem to be more subjective. The lowest α is observed for *emotional appeal* (0.26). The annotators most agreed on the *overall quality* ($\alpha = 0.51$), possibly meaning that the taxonomy adequately guides the assessment. In accordance with the moderate α -values, full agreement ranges between 17.4% and 44.7% only. On the contrary, we observe high majority agreement between 87.5% and 98% for all dimensions, even where scores are rather evenly distributed, such as for *global acceptability* (95.4%). In case of full disagreement, it makes sense to use score 2. We hence argue that the corpus is suitable for evaluating argumentation quality assessment.

Figure 2 shows all scores of each annotator for two example arguments from the corpus, referring to the question whether to ban plastic water bottles. Both have majority score 3 for *overall quality* (*Ov*),

Scores	Pro Water bottles, good or bad? Many people believe plastic water bottles to be good. But the truth is water bottles are polluting land and unnecessary. Plastic water bottles should only be used in emergency purposes only. The water in those plastic are only filtered tap water. In an emergency situation like Katrina no one had access to tap water. In a situation like this water bottles are good because it provides the people in need. Other than that water bottles should not be legal because it pollutes the land and big companies get 1000% of the profit.															Con Americans spend billions on bottled water every year. Banning their sale would greatly hurt an already struggling economy. In addition to the actual sale of water bottles, the plastics that they are made out of, and the advertising on both the bottles and packaging are also big business. In addition to this, compostable waters bottle are also coming onto the market, these can be used instead of plastics to eliminate that detriment. Moreover, bottled water not only has a cleaner safety record than municipal water, but it easier to trace when a potential health risk does occur. (http://www.friendsjournal.org/bottled-water) (http://www.cdc.gov/healthywater/drinking/bottled/)														
	Co	LA	LR	LS	Ef	Cr	Em	Cl	Ap	Ar	Re	GA	GR	GS	Ov	Co	LA	LR	LS	Ef	Cr	Em	Cl	Ap	Ar	Re	GA	GR	GS	Ov
Annotator A	3	3	3	2	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	
Annotator B	2	2	3	2	1	2	2	2	2	1	2	2	2	1	2	2	3	3	2	2	3	2	3	3	2	3	3	2	2	3
Annotator C	2	3	3	2	2	2	2	3	3	3	3	3	3	2	3	3	3	3	3	3	2	1	3	3	3	3	3	3	3	3
Majority score	2	3	3	2	2	2	2	3	3	3	3	3	3	2	3	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3

Figure 2: The scores of each annotator and the majority score for all considered quality dimensions of one pro and one con argument from our corpus. The arguments refer to the issue *ban plastic water bottles*.

but the pro argument shows more controversy with full disagreement in case of *effectiveness* (*Ef*). Especially, *annotator B* seems to be critical, giving one point less for several dimensions. In contrast, the con argument yields majority agreement for all 15 dimensions and full agreement for seven of them. It meets main quality criteria surveyed in Section 2, such as a rebuttal or references to resources. In fact, it constitutes the only corpus text with majority score 3 for *global sufficiency* (*GS*).

4.3 Correlations between Quality Dimensions

Table 3(c) compares the correlations of all dimension pairs. *Cogency* (.84), *effectiveness* (.81), and *reasonableness* (.86) correlate strongly with *overall quality*, and also much with each other.

Cogency and *local sufficiency* (.84) go hand in hand, whereas *local acceptability* and *local relevance* show the highest correlation with their global counterparts (.75 and .68 respectively). Quite intuitively, *credibility* and *appropriateness* correlate most with the acceptability dimensions. The coefficients of *emotional appeal* seem lower than expected, in particular for *effectiveness* (.31), indicating the limitation of a correlation analysis: As reflected by the 235 texts with majority score 2 for emotional appeal, many arguments make no use of emotions, thus obliterating effects of those which do. On the other hand, *clarity* was scored 2 in most cases, too, so the very low value there (.14) is more meaningful. Clarity rather correlates with *arrangement* (.56), which in turn shows coefficients above .50 for all high-level dimensions.

Altogether, the correlations largely match the surveyed theory. While an analysis of cause and effect should follow in future work, they provide first evidence for the adequacy of our taxonomy.

5 Conclusion

Argumentation quality is of high importance for argument mining, debating technologies, and similar. In computational linguistics, it has been treated only rudimentarily so far. This paper defines a common ground for the automatic assessment of argumentation quality in natural language. Based on a survey of existing theories and approaches, we have developed a taxonomy that unifies all major dimensions of logical, and dialectical argumentation quality. In addition, we freely provide an annotated corpus for studying these dimensions.

The taxonomy is meant to capture *all* aspects of argumentation quality, irrespective of how they can be operationalized. The varying inter-annotator agreement we obtained suggests that some quality dimensions are particularly subjective, raising the need to model the target audience of an argumentation. Still, the observed correlations between the dimensions support the general adequacy of our taxonomy. Moreover, most dimensions have already been approached on a certain abstraction level in previous work, as outlined. While some refinement may be suitable to meet all requirements of the community, we thus propose the taxonomy as the common ground for future research on computational argumentation quality assessment and the corpus as a first benchmark dataset for this purpose.

Acknowledgments

We thank all attendees of Dagstuhl Seminar 15512, particularly the rest of the quality breakout group: Wolf-Tilo Balke, Ruty Rinott, and Christian Stab. Also, we acknowledge financial support of the Stanford University, the DFG, and the Natural Sciences and Engineering Research Council of Canada.

References

- Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404. Association for Computational Linguistics.
- Maik Anderka, Benno Stein, and Nedim Lipka. 2012. Predicting quality flaws in user-generated content: The case of Wikipedia. In *Proceedings of the 35th International ACM Conference on Research and Development in Information Retrieval*, pages 981–990.
- Aristotle. 2007. *On Rhetoric: A Theory of Civic Discourse* (George A. Kennedy, Translator). Clarendon Aristotle series. Oxford University Press.
- Beata Beigman Klebanov, Christian Stab, Jill Burstein, Yi Song, Binod Gyawali, and Iryna Gurevych. 2016. Argumentation: Content, structure, and relationship with essay quality. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 70–75. Association for Computational Linguistics.
- Yonatan Bilu and Noam Slonim. 2016. Claim synthesis via predicate recycling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 525–530. Association for Computational Linguistics.
- J. Anthony Blair. 2012. *Groundwork in the Theory of Argumentation*. Springer Netherlands.
- Filip Boltužić and Jan Šnajder. 2015. Identifying prominent arguments in online debates using semantic textual similarity. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 110–115. Association for Computational Linguistics.
- Liora Braunstain, Oren Kurland, David Carmel, Idan Szpektor, and Anna Shtok. 2016. Supporting human answers for advice-seeking questions in CQA sites. In *Proceedings of the 38th European Conference on IR Research*, pages 129–141.
- Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, and Martin Chodorow. 1998. Enriching automated essay scoring using discourse marking. In *Discourse Relations and Discourse Markers*.
- Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 208–212. Association for Computational Linguistics.
- Daniel H. Cohen. 2001. Evaluating arguments and making meta-arguments. *Informal Logic*, 21(2):73–84.
- Richard Correnti, Lindsay Clare Matsumura, Laura Hamilton, and Elaine Wang. 2013. Assessing students’ skills at writing analytically in response to texts. *The Elementary School Journal*, 114(2):142–177.
- T. Edward Damer. 2009. *Attacking Faulty Reasoning: A Practical Guide to Fallacy-Free Arguments*. Wadsworth, Cengage Learning, Belmont, CA, 6th edition.
- Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357.
- Rory Duthie, Katarzyna Budynska, and Chris Reed. 2016. Mining ethos in political debate. In *Proceedings of the Sixth International Conference on Computational Models of Argument*, pages 299–310.
- Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996. Association for Computational Linguistics.
- Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 940–949. Dublin City University and Association for Computational Linguistics.
- Austin J. Freeley and David L. Steinberg. 2009. *Argumentation and Debate*. Cengage Learning, Boston, MA, 12th edition.
- James B. Freeman. 2011. *Argument Structure: Representation and Theory*. Springer.
- Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 549–554. Association for Computational Linguistics.
- Valentin Gold, Mennatallah El-Assady, Tina Bögel, Christian Rohrdantz, Miriam Butt, Katharina Holzinger, and Daniel Keim. 2015. Visual linguistic analysis of political discussions: Measuring deliberative quality. *Digital Scholarship in the Humanities*.
- Trudy Govier. 2010. *A Practical Study of Argument*. Wadsworth, Cengage Learning, Belmont, CA, 7th edition.
- Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. International corpus of learner English (version 2).

- Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics.
- Charles L. Hamblin. 1970. *Fallacies*. Methuen, London, UK.
- Hans Hoeken. 2001. Anecdotal, statistical, and causal evidence: Their perceived and actual persuasiveness. *Argumentation*, 15(4):425–437.
- Ralph H. Johnson and J. Anthony Blair. 2006. *Logical Self-defense*. International Debate Education Association.
- Klaus Krippendorff. 2007. Computing Krippendorff’s alpha reliability. Technical report, Univ. of Pennsylvania, Annenberg School for Communication.
- Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2008. Modeling and predicting the helpfulness of online reviews. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 443–452.
- Hugo Mercier and Dan Sperber. 2011. Why do humans reason? Arguments for an argumentative theory. *Behavioral and Brain Sciences*, 34:57–111.
- Nona Naderi and Graeme Hirst. 2015. Argumentation mining in parliamentary discourse. In *Principles and Practice of Multi-Agent Systems - International Workshops: IWEC 2014, Gold Coast, QLD, Australia, December 1-5, 2014, and CMNA XV and IWEC 2015, Bertinoro, Italy, October 26, 2015, Revised Selected Papers*, pages 16–25.
- Daniel J. O’Keefe and Sally Jackson. 1995. Argument quality and persuasive effects: A review of current approaches. In *Argumentation and Values: Proceedings of the Ninth Alta Conference on Argumentation*, pages 88–92.
- Nathan Ong, Diane Litman, and Alexandra Brusilovsky. 2014. Ontology-based argument mining and automatic essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 24–28. Association for Computational Linguistics.
- Myle Ott, Yejin Choi, Claire Cardie, and T. Jeffrey Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319. Association for Computational Linguistics.
- Joonsuk Park, Cheryl Blake, and Claire Cardie. 2015. Toward machine-assisted participation in eRulemaking: An argumentation model of evaluability. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, pages 206–210.
- Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948. Association for Computational Linguistics.
- Chaim Perelman, Lucie Olbrechts-Tyteca, John Wilkinson, and Purcell Weaver. 1969. *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame Press, Notre Dame, IN.
- Isaac Persing and Vincent Ng. 2013. Modeling thesis clarity in student essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 260–269. Association for Computational Linguistics.
- Isaac Persing and Vincent Ng. 2014. Modeling prompt adherence in student essays. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1534–1543. Association for Computational Linguistics.
- Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552. Association for Computational Linguistics.
- Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239. Association for Computational Linguistics.
- Niki Pfeifer, 2013. *Bayesian Argumentation: The Practical Side of Probability*, chapter On Argument Strength, pages 185–193. Springer Netherlands, Dordrecht.
- Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195. Association for Computational Linguistics.
- John L. Pollock. 2001. Defeasible reasoning with variable degrees of justification. *Artificial Intelligence*, 133(1–2):233–282.
- Zahra Rahimi, Diane J. Litman, Richard Correnti, Lindsay Clare Matsumura, Elaine Wang, and Zahid Kisa. 2014. Automatic scoring of an analytical response-to-text assessment. In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*, pages 601–610.
- Zahra Rahimi, Diane Litman, Elaine Wang, and Richard Correnti. 2015. Incorporating coherence of topics as a criterion in automatic response-to-text

- assessment of the organization of writing. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 20–30. Association for Computational Linguistics.
- Keith J. Ransom, Amy Perfors, and Daniel J. Navarro. 2015. Leaping to conclusions: Why premise relevance affects argument strength. *Cognitive Science*, pages 1–22.
- Ruty Rinott, Lena Dankin, Carlos Alzate Perez, M. Mitesh Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence — an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450. Association for Computational Linguistics.
- Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78. Association for Computational Linguistics.
- Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56. Association for Computational Linguistics.
- Christian Stab and Iryna Gurevych. 2016. Recognizing the absence of opposing arguments in persuasive essays. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 113–118. Association for Computational Linguistics.
- Christian Stab and Iryna Gurevych. 2017. Recognizing insufficiently supported arguments in argumentative essays. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.
- Marco R. Steenbergen, Andre Bachtiger, Markus Spornli, and Jürg Steiner. 2003. Measuring political deliberation: A discourse quality index. *Comparative European Politics*, 1:21–48.
- Reid Swanson, Brian Ecker, and Marilyn Walker. 2015. Argument mining: Extracting arguments from online dialogue. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 217–226. Association for Computational Linguistics.
- Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International World Wide Web Conference*, pages 613–624.
- Christopher W. Tindale. 2007. *Fallacies and Argument Appraisal. Critical Reasoning and Argumentation*. Cambridge University Press.
- Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.
- Frans H. van Eemeren and Rob Grootendorst. 2004. *A Systematic Theory of Argumentation: The Pragma-Dialectical Approach*. Cambridge University Press, Cambridge, UK.
- Frans H. van Eemeren. 2015. *Reasonableness and Effectiveness in Argumentative Discourse: Fifty Contributions to the Development of Pragma-Dialectics*. Argumentation Library. Springer International Publishing.
- Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment flow — A general model of web review argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 601–611. Association for Computational Linguistics.
- Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2016. Using argument mining to assess the argumentation quality of essays. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1680–1691. The COLING 2016 Organizing Committee.
- Henning Wachsmuth, Benno Stein, and Yamen Ajjour. 2017. “PageRank” for argument relevance. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.
- Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.
- Douglas Walton. 2006. *Fundamentals of Critical Argumentation*. Cambridge University Press.
- Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? Ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200. Association for Computational Linguistics.
- Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in oxford-style debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 136–141. Association for Computational Linguistics.
- Xiaoquan Zhao, Andrew Strasser, Joseph N. Cappella, Caryn Lerman, and Martin Fishbein. 2011. A measure of perceived argument strength: Reliability and validity. *Communication Methods and Measures*, 5(1):48–75.

A method for in-depth comparative evaluation: How (dis)similar are outputs of POS taggers, dependency parsers and coreference resolvers really?

Don Tuggener

University of Zurich

Institute of Computational Linguistics

tuggener@cl.uzh.ch

Abstract

This paper proposes a generic method for the comparative evaluation of system outputs. The approach is able to quantify the pairwise differences between two outputs and to unravel in detail what the differences consist of. We apply our approach to three tasks in Computational Linguistics, i.e. POS tagging, dependency parsing, and coreference resolution. We find that system outputs are more distinct than the (often) small differences in evaluation scores seem to suggest.

1 Introduction

While there exist well-defined procedures to evaluate system outputs against manually annotated gold data for many tasks in Computational Linguistics, generally little effort and exploration goes into identifying and analysing the differences between the outputs themselves. System outputs are usually compared in the following manner: The standard evaluation protocol for many tasks consists of comparing a system output (the *response*) to a manual annotation of the same data (the *key*). The difference between the response and key is quantified by a similarity metric such as accuracy, and different system outputs are compared to each other by ranking their scores with respect to the similarity metric.

However, comparing the scores of the similarity metric does not paint the full picture of the differences between the outputs, as we will demonstrate. There are hardly any principled or generic evaluation approaches that aim at comparing two or more system responses directly to investigate, highlight, and quantify their differences in detail. Closing this gap is desirable, because progress in many NLP tasks is often made in small steps, and it is

often left unclear what the specific contribution of a novel approach is if the comparison to related work is solely based on a (sometimes marginally) small improvement in F1 score or accuracy. Furthermore, an overall improvement regarding accuracy achieved by a new approach might come at the cost of failing in some areas where a baseline system was correct. Vice versa, a new approach might not improve overall accuracy, but solve particular problems that no other system has been able to address.

We propose an evaluation approach which aims at shedding light on the particular differences between system responses and which is intended as a complement to evaluation metrics such as F1-score and accuracy. By doing so, we strive to provide researchers with a tool that is able to give insight into the particular strengths and weaknesses of their system in comparison to others.¹ Our method is also useful in iterative system development, as it tracks changes in the outputs of different system versions or feature sets. Furthermore, our approach is able to compare multiple system outputs at once, which enables it to identify hard (or easy) problem areas by assessing how many of the systems solve a problem correctly and give according upper bounds for system ensembles. The performance difference between the simulated ensemble and the individual systems serves as an additional indicator of the difference between the system outputs.

We exemplify the application of our approach by aiming to answer the question of how (dis)similar are the outputs of several state-of-the-art systems for different tasks in NLP. We first motivate why using evaluation metrics such as accuracy is not suited for comparing outputs (next section). We then propose a method to do so which

¹Code available at: <https://github.com/dtuggener/ComparEval>

introduces an inventory to systematically classify and quantify output differences (section 2). Next, we demonstrate how combining a set of outputs can be used to identify their divergence and to identify hard (and easy) problem areas by looking at upper bounds in performance achieved by an oracle output combination (section 3).

1.1 Motivation

First let us motivate why comparing accuracy or F1 scores is not a suited method for establishing the (dis)similarity of system outputs. Consider a simple synthetic problem set with four test cases $\{A, B, C, D\}$ (e.g. a sequence of POS tags). A system response S_1 solves correctly the cases A and B , while a system response S_2 returns the correct answers for the cases C and D . In terms of accuracy, both responses achieve identical scores, i.e. 50%. However, their output is maximally dissimilar. Extending the set of cases, assume five problems, $\{A, B, C, D, E\}$, and three responses S_1, S_2 , and S_3 as shown in table 1. Although the three responses achieve the same accuracy (left table), their pairwise overlap in terms of identical correct responses (right table) varies considerably, i.e. S_1 is much more similar to S_2 (two shared answers) than to S_3 (one shared answer).

Key	A	B	C	D	E	Acc.	Overlap	
S_1	A	B	C	D	E	60%	S_1, S_2	67%
S_2	A	B	C	D	E	60%	S_2, S_3	67%
S_3	A	B	C	D	E	60%	S_1, S_3	33%

Table 1: Accuracy vs. Overlap on correct answers

In fact, the establishment of the similarity of the responses S_1, S_2 , and S_3 is more complicated, because we have left out the overlap of the incorrect answers in the responses. Consider the full responses in table 2.

Key	A	B	C	D	E	Acc.	Overlap	
S_1	A	B	C	X	Y	60%	S_1, S_2	40%
S_2	Z	B	C	D	U	60%	S_2, S_3	60%
S_3	Z	W	C	D	E	60%	S_1, S_3	20%

Table 2: Accuracy vs. Overlap on all answers

The overlap metric (right table) now compares how many of the cells in two rows have identical answers, regardless of whether the answer is correct. The overlap-based similarities between the systems have become more diverse, i.e. S_1 and S_3

are more dissimilar than in the previous table, and the similarities of the pairs (S_1, S_2) and (S_2, S_3) are now distinct, because S_2 and S_3 share the error Z (beside the correct answers C and D), while S_1 and S_2 do not share an error.

Hence, evaluating systems based on performance metrics such as accuracy and F1 scores provides no insight into the differences between the systems and is not able to accurately quantify the similarities between them. That is, a small difference in accuracy does not necessarily imply a high similarity of the outputs, and, vice versa, a larger difference in accuracy does not necessarily signify vastly dissimilar outputs.

Moreover, evaluation based on scores in performance metrics such as F1 does not detail in what regard a system performs better than another. Two systems might implement very distinct approaches, but achieve very similar scores in evaluation. Based on e.g. F1, we cannot assert whether a response S_2 performs better than a response S_1 because a) it solves the same problems as S_1 and then some additional ones, or b) if S_2 and S_1 solve a quite diverse set of problems and S_2 happens to solve a few more in its area of expertise. Additionally, a system that performs better than a baseline is bound to make errors where the baseline was correct. The overall accuracies cannot tell us how often this is the case.

In summary, the comparison of systems based on overall performance scores only lets us glimpse the proverbial tip of the iceberg. Therefore, our approach to comparative evaluation features three main points of interest:

1. How are the differences between system responses quantifiable?
2. What is the nature of the difference between two responses?
3. How can we assess the divergence of a set of responses, how complementary are they?

We try to answer these questions regarding three main tasks, namely POS tagging, dependency parsing, and coreference resolution. We select these tasks because they are fairly widespread procedures in Computational Linguistics and their evaluation increases in complexity. While we limit ourselves to these, we believe our approach to be generic enough to be applied to other labeling problems, such as named entity recognition and semantic role labeling.

2 Quantifying differences in system responses

As argued above, system responses differ both regarding the correct answers they give and the errors they make. The underlying idea of our approach is to assess how many of the labeled linguistic units (i.e. tokens) in the key have different labels in the responses, regardless of whether the labels are correct.² In a second step, we use a class inventory to analyse and quantify these differences in more detail.

Formally, given a set of tokens T and two accompanying system responses S_1 and S_2 , we quantify how many of the tokens $t_i \in T$ have a different label in S_1 and S_2 :

$$\text{diff}(S_1, S_2 | T) = \frac{|\forall t_i \in T : \text{label}(t_i, S_1) \neq \text{label}(t_i, S_2)|}{|T|} \quad (1)$$

Note that switching the inequality condition (\neq) to equality ($=$) actually yields the accuracy metric. That is, taking S_1 as the key and the S_2 as the response and calculating accuracy produces the inverted results of our metric, i.e. $1 - \text{diff}(S_1, S_2 | T)$, since accuracy is the ratio of tokens that have identical labels. The question is then, why not simply use S_1 as the key and S_2 as the response and calculate accuracy? While this answers whether two systems solve a similar or diverse set of problems, it does not enable us to identify the sources of the differences that drive the better performance of one response over the other. That is, if a token has a different label in S_1 and S_2 , we cannot tell which and if any of the responses is correct. Hence, we need to look at the gold labels of the tokens T in a key K . This enables us to categorise differences in the outputs into three distinct and informative classes³:

- **Correction:** S_1 labels a token incorrectly, S_2 corrects this error
- **New error:** S_1 is correct, S_2 introduces an error
- **Changed error:** Both S_1 and S_2 are incorrect but have different labels

The general algorithm to quantify differences in two responses S_1 and S_2 given a set of tokens $t_{i..n}$

²In other words, the complement of the overlap metric in tables 1 and 2.

³To motivate the nomenclature, we assume that S_1 is e.g. a baseline upon which S_2 tries to improve. However, the outputs can stem from any two systems.

in a key K is outlined in algorithm 1. This procedure lets us track and count how often S_2 has a different label than S_1 , classify the difference, and calculate the percentage of each class of difference. The approach can be applied straightforwardly to comparing outputs of POS taggers and dependency parsers.

Algorithm 1 Track differences in two responses

Input: Key $K \ni$ tokens $t_{i..n}$, Responses S_1, S_2

Output: Difference D , Changes C

```

1: for  $t_i \in K$  do
2:    $G = \text{label}(t_i, K)$ 
3:    $L_1 = \text{label}(t_i, S_1)$ 
4:    $L_2 = \text{label}(t_i, S_2)$ 
5:    $\text{TokCnt} + +$ 
6:   if  $L_1 \neq L_2$  then
7:     if  $L_2 = G$  then
8:        $C[\text{correction}][L_1, L_2] + +$ 
9:     else if  $L_1 = G$  then
10:       $C[\text{new error}][L_1, L_2] + +$ 
11:    else
12:       $C[\text{changed error}][L_1, L_2] + +$ 
13:       $\text{DiffLabel} + +$ 
14:  $D = \frac{\text{DiffLabel}}{\text{TokCnt}}$ 
15: return  $D, C$ 

```

2.1 POS tagging

We compare three POS taggers than can be used off-the-shelf to tag German: the Stanford POS Tagger (Toutanova et al., 2003), the TreeTagger (Schmid, 1995), and the Clevertagger (Sennrich et al., 2013, state-of-the-art). Following Sennrich et al. (2013), we use 3000 sentences from the TübaD/Z (Telljohann et al., 2004), a corpus of articles from a German newspaper, as a test set.⁴

Table 3 shows the labeling accuracy of the POS taggers and the percentage of correctly tagged sentences. The accuracy improvement of Clevertagger over TreeTagger is +1.27 points, and the percentage of correctly tagged sentences increases substantially (+9.9 points). In comparison to the Stanford tagger⁵, Clevertagger raises performance

⁴We change the POS tag for pronominal adverbs from PROP to PROAV in the test set, since all taggers feature only the latter tag.

⁵The most frequent error by the Stanford tagger is labeling some punctuation tokens (e.g. '-'') as '\$[' instead of '\$(' . Considering it a minor error, we replace all '\$[' labels in the Stanford response with '\$(', increasing accuracy from 86.60 to 90.41%.

	Accuracy	Correct sents.
Stanford	90.41	30.07
TreeTagger	94.89	46.87
Clevertagger	96.16	56.77

	<i>diff</i>	Δ Acc.
Stanford \leftrightarrow TreeTagger	11.06	4.48
Stanford \leftrightarrow Clevertagger	9.41	5.75
TreeTagger \leftrightarrow Clevertagger	4.96	1.27

Table 3: Accuracy and differences between POS taggers

by roughly 6 points in accuracy, but almost doubles the numbers of correctly tagged sentences.

In the lower table, we see that although the accuracy difference puts the Stanford tagger closer to the TreeTagger (4.48) than to the Clevertagger (5.75), the Stanford tagger’s response is more different from the one of TreeTagger (11.06) than from the response of Clevertagger (9.41). Comparing the two best performing taggers, we see that despite their accuracy difference of only 1.27 points, they label 4.96% of the tokens differently.

To get a more detailed understanding of the differences, we apply algorithm 1 to the two outputs, whose results are shown in table 4⁶, listing the five most frequent changes per difference class.⁷ Of the 4.96% different labels in Clevertagger compared to TreeTagger, 58.71% are corrections, 33.13% are new errors, and 8.15% changed errors.⁸ That is, one third of the changes that Clevertagger introduces are errors. This is a noteworthy observation which applies to all our system comparisons: Every improved response introduces a considerable amount of errors with respect to the baseline, i.e. it invalidates correct decisions of the baseline. While this observation is to some degree expected, our method is able to quantify and analyse such changes in detail.

Regarding the differences, we see that both the most frequent correction (NN→NE) and new error (NE→NN) evolve around the confusion of named entities and common nouns, which is especially

⁶For tag description see <http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html>

⁷Note that the change comparison can also be sorted by the biggest accuracy difference, cf. appendix table A.1.

⁸We here use the TreeTagger as S_1 and the Clevertagger as S_2 . Inverting the roles does not change the percentage values, but simply switches corrections to new errors and vice versa.

Difference: 4.96% (2674/53928)	
Corrections: 58.71% (1570/2674)	
NN→NE	27.96
PIAT→PIDAT	10.83
NN→ADJA	5.03
VVFIN→VVINF	4.52
NE→NN	3.25
New errors: 33.13% (886/2674)	
NE→NN	19.53
VVFIN→VVINF	9.48
NN→NE	9.26
ADV→ADJD	6.09
KOUS→PWAV	4.06
Changed errors: 8.15% (218/2674)	
NE→NN→FM	13.30
FM→NN→NE	7.34
NE→NN→ADJD	6.88
KOUS→KOKOM→PWAV	4.13
XY→NN→NE	2.29

Table 4: Token-based label changes comparing TreeTagger → Clevertagger (and Key → TreeTagger → Clevertagger for changed errors)

difficult for German, since capitalization cannot be exploited to distinguish the two. Furthermore, Clevertagger frequently invalidates TreeTagger’s correct labeling of finite verbs, tagging them as nonfinite (VVFIN → VVINF), although this change occurs under the most frequent corrections as well. While these are commonly known error sources for POS tagging German, our approach shows that they are in fact the main source of the differences between the output of the two top performing taggers.

2.2 Dependency parsing

The next task we investigate is dependency parsing. We choose English as the test language due to the lack of the availability of multiple parsers in other languages. We evaluate Google’s recently released Parsey McParseface (Andor et al., 2016, state-of-the-art) and two versions of the Stanford parser, i.e. the PCFG (Klein and Manning, 2003) and the Neural Network version (Chen and Manning, 2014). We follow the standard evaluation protocol and use section 23 of the PennTreebank (Marcus et al., 1993) as a test set and exclude punctuation tokens. We evaluate on Stanford Dependency labels (de Marneffe and Man-

ning, 2008), since Parsey support only them.⁹ We apply the parsers and their models "as is", i.e. we do not change any configuration settings.

	UAS	LS	LAS	Sent
Stan. PCFG	87.96	92.26	85.36	24.17
Stan. NN	88.68	92.45	86.43	26.95
Parsey	92.70	92.86	88.94	28.89

	<i>diff</i>	Δ LAS
Stan. PCFG \leftrightarrow Stan. NN	14.01	1.07
Stan. PCFG \leftrightarrow Parsey	15.49	3.58
Stan. NN \leftrightarrow Parsey	13.62	2.51

Table 5: Parser performance and difference

In table 5, we report the unlabeled attachment score (UAS), the labeling score (LS), and the labeled attachment score (LAS) for the parsers. Furthermore, we evaluate how many of the sentences are fully parsed correctly given each criterion.

We see that Parsey outperforms the Stanford parsers mainly due to the performance in attachment (UAS). The performance differences in assigning grammatical labels (LS) are comparably marginal. Parsey also features almost identical performance in both attaching and labeling tokens. However, there is a gap compared to labeled attachment score, which indicates that although Parsey attaches more tokens correctly than the other parsers, it does not necessarily assign the correct grammatical label to these tokens. Looking at the difference chart, we see that despite the rather small differences in LAS (1-4 points), the parsers attach and label around 15% of the tokens differently. The Stanford parsers only differ in 1.07 points in LAS, but this difference is based on 14.01% (*diff*) of the tokens in the test set. Parsey outperforms the Stanford NN parser by 2.51 LAS based on 13.62% of the tokens. To gain a better understanding of the differences contained in these 13.62% of the tokens, we apply algorithm 1, whose output is shown in table 6.

The table shows that half (50.22%) of the 13.62% changed token annotations from Stanford NN to Parsey are corrections. All of these changes are attachment corrections, i.e. the label of the tokens are not changed, which correlates with the small difference we saw in labeling score. The

⁹We convert the PennTreebank to Stanford dependencies using the Penn Treebank converter included in the Stanford parser (<http://nlp.stanford.edu/software/stanford-dependencies.shtml>).

Difference: 13.62% (6776/49748)

Corrections: 50.22% (3403/6776)

nn \rightarrow nn	10.93
prep \rightarrow prep	9.49
cc \rightarrow cc	5.32
conj \rightarrow conj	4.17
advmod \rightarrow advmod	2.59

New errors: 31.79% (2154/6776)

vmod \rightarrow partmod	9.38
amod \rightarrow nn	8.08
prep \rightarrow prep	7.38
vmod \rightarrow infmod	5.43
npadvmod \rightarrow dep	4.32

Changed errors: 17.99% (1219/6776)

prep \rightarrow prep \rightarrow prep	5.00
vmod \rightarrow vmod \rightarrow partmod	2.95
advmod \rightarrow advmod \rightarrow advmod	1.97
cc \rightarrow cc \rightarrow cc	1.89
conj \rightarrow conj \rightarrow conj	1.23

Table 6: Token-based analysis of parser differences (Stanford NN \rightarrow Parsey, LAS)

two most prominent corrections are the attachment of noun compound modifiers (nn) and prepositions (prep). Almost one third (31.79%) of the changes are new errors. Compared to the corrections, Parsey here changes the labels of the tokens. Contrarily, changed errors (17.99%) constitute roughly one fifth of the changes and mainly consist of changes in attachment.

2.3 Coreference resolution

The final task we investigate is coreference resolution. We choose three freely available systems for English, again due to the lack of available systems for other languages: the Stanford statistical coreference resolver (Clark and Manning, 2015, state-of-the-art), HOTCoref (Björkelund and Kuhn, 2014), and the Berkeley coreference system (Durrett and Klein, 2013). We use the CoNLL 2012 shared task test set (Pradhan et al., 2012).

The coreference task differs from the previous two, since not all tokens in a document partake in coreference relations (but all tokens are in syntactic relations and feature a POS tag). Furthermore, the linguistic units of coreference relations are not only single word tokens, but syntactic units called *mentions* (i.e. mostly noun phrases). Therefore, we have to adapt our similarity metric in equation 1. To quantify the difference of two corefer-

ence system outputs S_1 and S_2 , given a key K , we count how many of the mentions m are classified differently using a mention classification function c :

$$\text{diff}(S_1, S_2 | K) = \frac{|\forall m \in S_1 \cap S_2 \cap K : c(m, S_1) \neq c(m, S_2)|}{|\forall m \in S_1 \cap S_2 \cap K|} \quad (2)$$

The mention classification function c requires a class inventory which is not featured by the common evaluation metrics for coreference resolution.¹⁰ Therefore, we adapt the mention classification paradigm introduced in the ARCS framework for coreference resolution evaluation (Tuggener, 2014) which assigns one of the following four classes to a mention m given a key K and a system response S :

- True Positive (TP): m is correctly resolved to an antecedent.
- False Positive (FP): m has no antecedent in K but one in S .
- False Negative (FN): m has no antecedent in S but one in K .
- Wrong Linkage (WL): m has an antecedent in K but is assigned an incorrect antecedent in S .

However, one issue with ARCS is to determine a criterion for the TP class, i.e. under what circumstances is m regarded as resolved correctly. Tuggener (2014) proposed to determine correct antecedents based on the requirements of prospective downstream applications.¹¹ We implement one loose criterion and regard m as correctly resolved if any of its antecedents in S is also an antecedent of m in K . Conversely, if none of the antecedents of m overlap in S and K , we label m as WL. This yields the ARCS_{any} metric. Alternatively, we require that the closest preceding nominal antecedent of m in S is also an antecedent of m in K , which yields the ARCS_{nom} metric. This metric is more conservative in assigning the TP class, but implements a more realistic criterion for

¹⁰The common metrics analyse either the links between mentions or calculate a percentage of overlapping mentions in coreference chains in the key and a response. They are not able to determine whether a given mention m is resolved correctly or assign a class to it.

¹¹Machine translation requires pronouns to be linked to nominal antecedents, Sentiment analysis needs Named Entity antecedents (if available) etc.

correct antecedents from the perspective of downstream applications.

The official CoNLL score MELA (average of MUC, CEAFE, and BCUB) and the recently proposed LEA metric (Moosavi and Strube, 2016), which addresses several issues of the other metrics, as well as the ARCS scores, are given in table 7. Using the ARCS class inventory and equation 2, we quantify how many of the mentions are classified differently in the system responses.

	MELA	LEA	ARCS_{any}	ARCS_{nom}
Berkeley	62.06	54.80	71.25	59.13
HOTCoref	64.32	57.13	73.27	61.95
Stanford	66.62	60.92	76.30	62.04

ARCS_{any}	diff	ΔF1
Stanford \leftrightarrow Berkeley	27.13	5.05
Stanford \leftrightarrow HOTCoref	26.30	3.03
Berkeley \leftrightarrow HOTCoref	27.22	2.02

ARCS_{nom}	diff	ΔF1
Stanford \leftrightarrow Berkeley	35.39	2.91
Stanford \leftrightarrow HOTCoref	37.45	0.09
Berkeley \leftrightarrow HOTCoref	34.89	2.82

Table 7: Coreference resolution evaluation (F1) and differences (%)

The F1 scores are lowest for the LEA metric, because it gives more weight to errors regarding longer coreference chains. The ARCS_{any} metric assigns the highest scores due to the loose criterion that any antecedent is correct as long as it is in the key chain of a given mention. Furthermore, all the metrics agree on the ranking of the systems.

The mention-based differences between the systems are considerably larger than the relatively small differences in F1 scores suggest. The Stanford system outperforms HOTCoref by 2.3 MELA, 3.79 LEA F1, and 3.03 ARCS_{any} F1, but the systems process one fourth (26.30%) of the mentions differently in the ARCS_{any} setting. For the ARCS_{nom} criterion, the differences are even larger. The Stanford system outperforms the Berkeley system by 2.91 ARCS_{nom} F1, but the systems process 35.39% of the mentions differently. Furthermore, we observe that the differences in F1 (ΔF1) do not correlate with the differences of the outputs (diff) for both ARCS metrics. Given ARCS_{nom} , we see that the smallest difference in F1 (Stanford \leftrightarrow HOTCoref: 0.09) actually occurs between the two responses that the diff metric deems most dissimilar (37.45).

Finally, we apply algorithm 1, using the ARCS_{nom} criterion and our mention classification

scheme to the two best performing systems, i.e. HOTCoref and Stanford. Results are given in table 8.

Difference: 37.45% (5760 / 15382)	
Corrections: 44.62% (2570/5760)	
wl → tp	20.09
fn → tp	12.34
fp → tn	12.19
New errors: 41.65% (2399/5760)	
tp → wl	17.08
tp → fn	12.33
tn → fp	12.24
Changed errors: 13.73% (791/5760)	
fn → wl	3.87
wl → fn	9.86

Table 8: Comparison of two coreference responses (HOTCoref → Stanford)

We see that less than 50% of the changes that the Stanford system introduces are corrections (44.62%). But this percentage is still higher than the newly introduced errors (41.65%); hence the improvement in overall F1. Furthermore, the most frequent change is wrong linkages to true positives (wl → tp). The most frequent new error also involves true mentions, i.e. attaching correctly resolved mentions to incorrect antecedents (tp → wl). Recovering false negatives and rendering true positives to false negatives occurs equally frequent, roughly. Hence, the performance difference stems mainly from attaching anaphoric mention to (nominal) antecedents, rather than from deciding which mentions to resolve, which are two sub-problems in coreference resolution.

3 System combination

Lastly, we combine the system outputs per task and calculate the upper bounds for perfect system combinations by deeming a token labeled correctly if at least one of the systems provides the correct label. The upper bounds are intended to be another measure of the (dis)similarity of the outputs: the higher the upper bound, the higher the divergence of the outputs. Furthermore, looking at per-label performance of all systems, we can identify labels with low scores but high upper bounds, which is an interesting starting point for future work.

3.1 POS tagging

We start with the POS tagging task and present the upper bound of the system combination in table 9. We also indicate the accuracy gains for the top ten most frequent POS tags relative to the best performing tagger (Clevertagger).

	Stan.	Tree.	Clever.	Upper bound	
Overall	90.41	94.38	96.16	98.52	+2.36
NN	96.01	98.47	98.06	99.55	+1.49
ART	99.62	99.32	99.43	99.85	+0.42
APPR	86.10	98.03	99.20	99.56	+0.36
NE	87.35	77.46	85.31	95.17	+9.86
ADJA	92.73	94.50	98.40	99.44	+1.04
ADV	89.25	91.71	90.93	95.48	+4.55
VVFIN	79.73	95.15	91.52	97.48	+5.96
VVAFIN	91.97	98.93	97.74	99.56	+1.82
KON	97.13	95.37	96.55	98.37	+1.82
ADJD	72.37	89.29	88.80	93.53	+4.73

Table 9: POS tagging upper bounds and accuracy (highest scores in green; lowest in red; middle in yellow)

The Stanford tagger, despite performing the lowest with respect to overall accuracy, achieves the highest accuracy on named entities (NE), while the TreeTagger struggles in this category particularly. The TreeTagger surpasses the other taggers on finite verbs (VVFIN) by a wide margin and auxiliary finite verbs (VAFIN). Clevertagger performs best overall, but interestingly, it only achieves the highest accuracy on three of the ten most frequent POS tags. Looking at the overall upper bound, we see that it more than halves the error rate of the best performing system and is near 99% accuracy. The POS tags that profit most from the combination are named entities. Interestingly, all the taggers have low accuracy with respect to this tag, but the upper bound of the combination drastically raises it. Hence, it seems that the taggers diverge mostly here, which correlates with our analysis of the difference between the two best performing systems in table 4.

3.2 Dependency parsing

Next, we analyse the upper bounds of the combination of the dependency parsers, given in table 10. In contrast to the POS tagging task, we find that the best performing system, Parsey (P-MP) achieves highest LAS for almost all considered labels. Still, its overall LAS is drastically increased by the upper bound (+5.99) of the perfect system combination. Two of the labels that benefit the most of the combination are amod (adjec-

tival modifier), which is often confused with nn (noun compound modifier) as we saw in table 6, and advmod (adverb modifier). All parsers have below 90 LAS for these labels, but the combination raises performance to 95.26 and 91.40, respectively. Furthermore, prepositions (prep) gains considerably in LAS in the combination. We observed in table 6 that almost ten percent of the difference between Parsey and the Stanford NN parser stem from correcting attachments of prepositions. However, also more than seven percent of the difference stems from invalidating correctly attached prepositions in the Stanford NN output. The large performance jump in the combination of the systems is further evidence that the parsers are highly complementary with respect to prepositions.

	S-PCFG	S-NN	P-MP	Upper bound	
All	85.36	86.43	88.94	94.93	+5.99
prep	78.76	84.26	88.21	94.18	+5.97
pobj	94.26	95.30	96.35	98.62	+2.27
det	96.81	96.65	98.66	99.38	+0.72
nn	74.64	76.81	86.36	88.91	+2.55
nsubj	92.08	89.78	94.41	97.85	+3.44
amod	87.59	88.45	86.95	95.26	+8.31
root	93.79	89.67	95.74	98.63	+2.89
doj	90.19	90.88	92.91	97.47	+4.56
aux	97.53	97.11	97.63	99.30	+1.67
advmod	74.48	78.56	82.97	91.40	+8.43

Table 10: Parsing accuracy (LAS) and upper bounds.

3.3 Coreference resolution

For the coreference task, it is not trivial to calculate the F1 upper bound of the response combination, as the systems do not feature the same mentions in their outputs¹², and disentangling the false positives is a cumbersome undertaking. Therefore, we limit our investigation to the gold mentions in the key and count for how many of them at least one of the responses produces a correct nominal antecedent, which yields the upper bound for $ARCS_{nom}$ recall. To gain a deeper insight into the benefits of the combination and the performance of the systems, we divide the mentions into nouns (named entities and common nouns), personal pronouns (PRP), and possessive pronouns (PRP\$). Results are given in table 11.¹³

The system with overall best recall features the highest recall with respect to all mention types.

¹²The systems have to decide which NPs they consider for coreference resolution (the anaphoricity detection problem). I.e. the mentions are not known beforehand, and the systems

	Berk.	Stan.	HOT.	Upper bound	
Overall	55.72	58.13	59.34	73.39	+14.05
Nouns	59.67	59.76	60.99	73.48	+12.49
PRP	50.66	56.30	57.56	73.80	+16.24
PRP\$	62.36	64.62	65.98	80.57	+14.59

Table 11: Mention-based coreference performance ($ARCS_{nom}$ recall) and upper bounds

However, there is a considerable difference in recall to the mention types for all systems: Possessive pronouns are more easily attached to correct nominal antecedents than nouns and personal pronouns. Furthermore, we see that upper bounds raise recall uniformly for all mention types by a considerable margin. This suggests that the outputs are indeed different in several regards, which correlates to the comparisons in tables 7 and 13.

4 Related work

One way to establish the difference of two system outputs is to apply statistical significance tests. However, there is generally little agreement on which test to use, and it is often not trivial to verify if all criteria are met for the application of a specific test to a given data set (Yeh, 2000). Furthermore, the significance tests provide no insight into the nature of the differences between two outputs.

Several survey papers analysed performance of state-of-the-art tools for POS tagging (Volk and Schneider, 1998; Giesbrecht and Evert, 2009; Horsmann et al., 2015) or dependency parsing (McDonald and Nivre, 2007). While these surveys provide performance results along different axes (accuracy, time, domain, frequent errors), they do not analyse the particular differences between the system responses on the token level and hence do not provide a (dis)similarity rating of the responses. Regarding dependency parsing, our work is most closely related to McDonald and Nivre (2007) and Seddah et al. (2013). Both papers analyse the performance of parsers with respect to several subproblems. McDonald and Nivre (2007) also performed output combination experiments to stress that the two parsers that they investigated are complementary to a significant degree.

Comparative system evaluation in shared tasks is usually performed by pitting scores in evalua-

will hallucinate different incorrect ones.

¹³Note that the HOTCoref system has better recall than the Stanford system, but the Stanford system features better precision, which leads to a higher F1 score in table 7.

tion metrics against each other, e.g. the CoNLL shared tasks on coreference (Pradhan et al., 2011; Pradhan et al., 2012) or on dependency parsing (Buchholz and Marsi, 2006; Nilsson et al., 2007). While the post task evaluation of the CoNLL shared task 2007 included an experiment of system combination which showed performance improvements, it is generally left unclear how similar are the system outputs with (sometimes marginally) small differences with respect to the evaluation metrics.

Another branch of evaluation related to our work is error analysis. Gärtner et al. (2014) presented a tool to explore coreference errors visually, but does not aggregate and classify them. Kummerfeld and Klein (2013) devised a set of error classes for coreference and analysed quantitatively which systems make which errors. Martschat and Strube (2014) presented an analysis and grouping of recall errors for coreference and evaluated a set of system responses. However, these analyses focus on the errors of one system at a time and then compare the overall error statistics, i.e. there is no direct linking or combination of the responses. Hence, we believe our approach to be complementary to the work outlined above.

5 Conclusion

We have presented a generic dissimilarity metric for system outputs and applied it to several systems for POS tagging, dependency parsing, and coreference resolution. We found that systems with marginal differences in accuracy scores or F1 actually have considerably distinct outputs. We combined system outputs and calculated upper bounds in performance as an additional measure of the degree of difference between the outputs.

We discussed and applied a method for analysing the specific differences between two system outputs using a class inventory to label and quantify the differences. Our analysis revealed the (often considerable) quantity of new errors that improvements introduce compared to baselines. We believe that this kind of analysis is also useful during system and method design, as it allows one to track all changes in the output when adjusting a system or a feature set.

While we have explored our approach on three core tasks in Computational Linguistics, we believe it to be applicable to other areas in the field. Our hope is that our method of comparative evalu-

ation will motivate other researchers to gain an in-depth understanding of the output of their systems and what distinguishes them from others, beyond differences in accuracy or F1 scores.

References

- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Google.
- Anders Björkelund and Jonas Kuhn. 2014. Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57, Baltimore.
- Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X '06*, pages 149–164.
- Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.
- Kevin Clark and Christopher D Manning. 2015. Entity-Centric Coreference Resolution with Model Stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1415, Beijing, China.
- Marie-Catherine de Marneffe and Christopher D Manning. 2008. The Stanford Typed Dependencies Representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.
- Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA.
- Markus Gärtner, Anders Björkelund, Gregor Thiele, Wolfgang Seeker, and Jonas Kuhn. 2014. Visualization, Search, and Error Analysis for Coreference Annotations. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 7–12, Baltimore, Maryland.

- Eugenie Giesbrecht and Stefan Evert. 2009. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. In *Proceedings of the 5th Web as Corpus Workshop (WAC5)*, San Sebastian, Spain.
- Tobias Horstmann, Nicolai Erbs, and Torsten Zesch. 2015. Fast or Accurate? - A Comparative Evaluation of PoS Tagging Models. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL-2015)*, Essen, Germany.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 423–430.
- JK Jonathan K Kummerfeld and Dan Klein. 2013. Error-Driven Analysis of Challenges in Coreference Resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 265–277, Seattle.
- Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, jun.
- Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 2070–2081, Doha.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.
- Nafise Sadat Moosavi and Michael Strube. 2016. Which Coreference Evaluation Metric Do You Trust? A Proposal for a Link-based Entity Aware Metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1.
- Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932.
- Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–27, Portland.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning*, Jeju.
- Helmut Schmid. 1995. Improvements In Part-of-Speech Tagging With an Application To German. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.
- Djame Seddah, Reut Tsarfaty, Sandra Kuebler, Marie Candito, Jinho Choi, Richard Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Wolin ski, Alina Wroblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*.
- Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 601–609, Hissar.
- Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. 2004. The Tüba-D/Z Treebank: Annotating German with a Context-Free Backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 2229–2232, Lisbon.
- Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 173–180.
- Don Tuggener. 2014. Coreference Resolution Evaluation for Higher Level Applications. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 231–235, Gothenburg.
- Martin Volk and Gerold Schneider. 1998. Comparing a statistical and a rule-based tagger for German. In *Proceedings of KONVENS*.
- Alexander Yeh. 2000. More Accurate Tests for the Statistical Significance of Result Differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, pages 947–953.

A Appendix

A.1 POS tagging

Δ Acc.	TT	CT	POS	#tok
-100.00	100.00	0.00	VAIMP	1
+97.34	0.00	97.34	PROAV	301
+96.65	0.00	96.65	PIDAT	179
+50.00	14.74	64.74	FM	156
+46.67	46.67	93.33	PWAT	15
+39.39	54.55	93.94	PTKA	33
+33.33	33.33	66.67	VVIMP	6
+28.57	71.43	100.00	APZR	7
+26.87	71.64	98.51	PWAV	67
-24.00	96.00	72.00	VMINF	25
+13.33	78.33	91.67	KOUI	60

Table 12: Largest accuracy differences between TreeTagger (TT) and Clevertagger(CT); number of token with POS tag in the test set (#tok)

A.2 Coreference resolution

Since the ARCS framework is relatively unknown and not widely used, we revisit the connection of our *diff* metric to accuracy and F1 outlined in section 2 in order to use one of the coreference metrics to establish the differences between the outputs. We saw that our metric is inversely equivalent to accuracy when taking one system response as the key and the other as the response. That is, we can calculate the *diff* ratio by $1 - \frac{|t_i \in T: label(t_i, S_1) \neq label(t_i, S_2)|}{|T|}$, which is equivalent to taking S_1 as the key and S_2 as the response (or vice versa). For the coreference task, we can thus use one response as the key and the other as the response. The resulting F1 score can then be used as an agreement value, which, however, does not provide any detailed analysis of the nature of the differences compared to the ARCS approach. Table 13 shows the F1 scores when using one response as the key and the second as response. Note that switching the key and the response role provides the same F1 scores for two responses; the only effect is that the recall and precision values are switched.

The table shows that using this approach, we obtain F1 scores that give quite high dissimilarities when turned into the *diff* metric, i.e. $diff = 100 - F1$. The average of the *diff* metric given MELA F1 is 28.90 ($100 - 71.10$); given LEA F1 it is 35.22 ($100 - 64.78$). Compared to the ARCS_{any}

Key	Response	LEA F1	MELA F1
Berkeley	HOTCoref	63.58	70.32
Berkeley	Stanford	66.03	71.91
Stanford	HOTCoref	64.73	71.08

Table 13: Coreference system comparison pairing responses

average *diff* (25.56) and the ARCS_{nom} average *diff*, 34.09, the values are in a similar range.

Re-evaluating Automatic Metrics for Image Captioning

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem

Hacettepe University Computer Vision Lab

Dept. of Computer Engineering, Hacettepe University, Ankara, TURKEY

kilickayamert@gmail.com, {aykut, nazli, erkut}@cs.hacettepe.edu.tr

Abstract

The task of generating natural language descriptions from images has received a lot of attention in recent years. Consequently, it is becoming increasingly important to evaluate such image captioning approaches in an automatic manner. In this paper, we provide an in-depth evaluation of the existing image captioning metrics through a series of carefully designed experiments. Moreover, we explore the utilization of the recently proposed Word Mover's Distance (WMD) document metric for the purpose of image captioning. Our findings outline the differences and/or similarities between metrics and their relative robustness by means of extensive correlation, accuracy and distraction based evaluations. Our results also demonstrate that WMD provides strong advantages over other metrics.

1 Introduction

There has been a growing interest in research on integrating vision and language in natural language processing and computer vision communities. As one of the key problems in this emerging area, image captioning aims at generating natural descriptions of a given image (Bernardi et al., 2016). This is a challenging problem since it requires the ability to not only understand the visual content, but also to generate a linguistic description of that content. In this regard, it can be framed as a machine translation task where the source language denotes the visual domain and the target language is a specific language such as English. The recently proposed deep image captioning studies follow this interpretation and model the process

via an encoder-decoder architecture (Vinyals et al., 2015; Xu et al., 2015; Karpathy and Fei-Fei, 2015; Jia et al., 2015). These approaches have attained considerable success in the recent benchmarks such as FLICKR8K (Hodosh et al., 2013), FLICKR30K (Young et al., 2014) and MS COCO (Lin et al., 2014) as compared to the earlier techniques which explicitly detect objects and generate descriptions by using surface realization techniques (Kulkarni et al., 2013; Li et al., 2011; Elliott and Keller, 2013).

With the size of the benchmark datasets becoming larger and larger, evaluating image captioning models has become increasingly important. Human-based evaluations become obsolete as they are costly to acquire and, more importantly, not repeatable. Automatic evaluation metrics are employed as an alternative to human evaluation in both developing new models and comparing them against the state-of-the-art. These metrics compute a score that indicates the similarity/dissimilarity between an automatically generated caption and a number of human-written reference (gold standard) descriptions.

Some of these automatic metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and TER (Snover et al., 2006) have originated from the readily available metrics for machine translation and/or text summarization. On the contrary, the more recent metrics such as CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016) are specifically developed for image caption evaluation task.

Evaluation with automatic metrics has some challenges as well. As previously analyzed in (Elliott and Keller, 2014), the existing automatic evaluation measures have proven to be inadequate in successfully mimicking the human judgements for evaluating the image descriptions. The latest eval-

Table 1: A summary of the evaluation metrics considered in this study.

Metric	Proposed to evaluate	Underlying idea
BLEU (Papineni et al., 2002)	Machine translation	n -gram precision
ROUGE (Lin, 2004)	Document summarization	n -gram recall
METEOR (Banerjee and Lavie, 2005)	Machine translation	n -gram with synonym matching
CIDEr (Vedantam et al., 2015)	Image description generation	<i>tf-idf</i> weighted n -gram similarity
SPICE (Anderson et al., 2016)	Image description generation	Scene-graph synonym matching
WMD (Kusner et al., 2015)	Document similarity	Earth Mover Distance on <i>word2vec</i>

uation results of 2015 MS COCO Challenge on image captioning has also revealed some interesting findings in line with this observation (Vinyals et al., 2016). In the challenge, the recent deep models outperform the human upper bound according to automatic measures, yet they could not beat the humans when the subjective human judgements are considered. These demonstrate that we need to better understand the drawbacks of existing automatic evaluation metrics. This motivates us to present an in-depth analysis of the current metrics employed in image description evaluation.

We first review BLEU, ROUGE, METEOR, CIDEr and SPICE metrics, and discuss their main drawbacks. In this context, we additionally describe WMD metric which has been recently proposed as a distance measure between text documents in (Kusner et al., 2015). We then investigate the performance of these automatic metrics through different experiments. We analyze how well these metrics mimic human assessments by estimating their correlations with the collected human judgements. Different from the previous related work (Elliott and Keller, 2014; Vedantam et al., 2015; Anderson et al., 2016), we perform a more accurate analysis by additionally reporting the results of Williams significance test. This further allows us to figure out the differences and/or similarities between a pair of metrics, whether any two metrics complement each other or provide similar results. We then test the ability of these metrics to distinguish certain pairs of captions from one another in reference to a ground truth caption. Next, we carry out an analysis on robustness of these metrics by analyzing how well they cope with the distractions in the descriptions (Hodosh and Hockenmaier, 2016).

2 Evaluation Metrics

A summary of the metrics investigated in our study is given in Table 1. All these metrics ex-

cept SPICE and WMD define the similarity over words or n -grams of reference and candidate descriptions by considering different formulas. On the other hand, SPICE (Anderson et al., 2016) considers a scene-graph representation of an image by encoding objects, their attributes and relations between them, and WMD leverages word embeddings to match groundtruth descriptions with generated captions.

2.1 BLEU

BLEU (Papineni et al., 2002) is one of the first metrics that have been in use for measuring similarity between two sentences. It has been initially proposed for machine translation, and defined as the geometric mean of n -gram precision scores multiplied by a brevity penalty for short sentences. In our experiments, we use the smoothed version of BLEU as described in (Lin and Och, 2004).

2.2 ROUGE

ROUGE (Lin, 2004) is initially proposed for evaluation of summarization systems, and this evaluation is done via comparing overlapping n -grams, word sequences and word pairs. In this study, we use ROUGE-L version, which basically measures the longest common subsequences between a pair of sentences. Since ROUGE metric relies highly on recall, it favors long sentences, as also noted by (Vedantam et al., 2015).

2.3 METEOR

METEOR (Banerjee and Lavie, 2005) is another machine translation metric. It is defined as the harmonic mean of precision and recall of uni-gram matches between sentences. Additionally, it makes use of synonyms and paraphrase matching. METEOR addresses several deficiencies of BLEU such as recall evaluation and the lack of explicit word matching. n -gram based measures work reasonably well when there is a significant overlap

between reference and candidate sentences; however they fail to spot semantic similarity when the common words are scarce. METEOR handles this issue to some extent using WordNet-based synonym matching, however just looking at synonyms may be too restrictive to capture overall semantic similarity.

2.4 CIDER

CIDER (Vedantam et al., 2015) is a recent metric proposed for evaluating the quality of image descriptions. It measures the consensus between candidate image description c_i and the reference sentences, which is a set $S_i = \{s_{i1}, \dots, s_{im}\}$ provided by human annotators. For calculating this metric, an initial stemming is applied and each sentence is represented with a set of 1-4 grams. Then, the co-occurrences of n -grams in the reference sentences and candidate sentence are calculated. In CIDER, similar to *tf-idf*, the n -grams that are common in all image descriptions are down-weighted. Finally, the cosine similarity between n -grams (referred as $CIDER_n$) of the candidate and the references is computed.

CIDER is designed as a specialized metric for image captioning evaluation, however, it works in a purely linguistic manner, and only extends existing metrics with *tf-idf* weighting over n -grams. This sometimes causes unimportant details of a sentence to be weighted more, resulting in a relatively ineffective caption evaluation.

2.5 SPICE

Another recently proposed metric for evaluating image caption similarity is SPICE (Anderson et al., 2016). It is based on the agreement of the scene-graph tuples (Johnson et al., 2015; Schuster et al., 2015) of the candidate sentence and all reference sentences. Scene-graph is essentially a semantic representation that parses the given sentence to semantic tokens such as object classes C , relation types R and attribute types A . Formally, a candidate caption c is parsed into a scene-graph as

$$G(c) = \langle O(c), E(c), K(c) \rangle$$

where $G(c)$ denotes the scene graph of caption c , $O(c) \subseteq C$ is the set of object mentions, $E(c) \subseteq O(c) \times R \times O(c)$ is the set of hyper-edges representing relations between objects, and $K(c) \subseteq O(c) \times A$ is the set of attributes associated with objects. Once the parsing is done, a set of tuples is

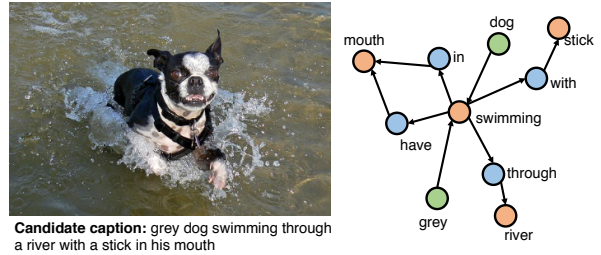


Figure 1: An example image with its Scene Graph where the parser fails to parse the candidate sentence accurately, which could result in wrong calculation of SPICE metric. See text for details.

formed by using the elements of G and their possible combinations. SPICE score is then defined as the F_1 -score based on the agreement between the candidate and reference caption tuples. For tuple matching, SPICE uses WordNet synonym matching (Pedersen et al., 2004) as in METEOR (Banerjee and Lavie, 2005). One problem is that the performance becomes quite dependent on the quality of the parsing. Figure 1 illustrates an example case of failure. Here, swimming is parsed as an object, with all its relations, and dog is parsed as an attribute.

2.6 WMD

Two captions may not share the same words or any synonyms; yet they can be semantically similar. On the contrary, two captions may include similar objects, attributes or relations yet they may not be semantically similar. Metrics that are currently in use fail to correctly identify and assess the quality of such cases. To address this issue, we propose to use a recently introduced document distance measure called Word Mover’s Distance (WMD) (Kusner et al., 2015) for evaluating image captioning approaches. WMD casts the distance between documents as an instance of Earth Mover’s Distance (EMD) (Rubner et al., 2000), where travel costs are calculated based on *word2vec* (Mikolov et al., 2013) embeddings of the words.

For WMD, text documents (in our case image captions) are first represented by their normalized bag-of-words (nBOW) vectors, accounting for all words except stopwords. More formally, each text document is represented as vectors $\mathbf{d} \in R^n$, where, $d_i = \frac{c_i}{\sum_{j=1}^n c_j}$ if a word i appears c_i times in the document. WMD incorporates semantic similarity between individual word pairs into the document similarity metric, by using the distances in

Table 2: Drawbacks of automatic evaluation metrics for image captioning. See text for details.

	Description	BLEU	METEOR	ROUGE	CIDEF	SPICE	WMD
original	a man wearing a red life jacket is sitting in a canoe on a lake	1	1	1	10	1	1
candidate	a man wearing a life jacket is in a small boat on a lake	0.45	0.28	0.67	2.19	0.40	0.19
synonyms	a guy wearing a life vest is in a small boat on a lake	0.20	0.17	0.57	0.65	0.00	0.10
redundancy	a man wearing a life jacket is in a small boat on a lake at sunset	0.45	0.28	0.66	2.01	0.36	0.18
word order	in a small boat on a lake a man is wearing a life jacket	0.26	0.26	0.38	1.32	0.40	0.19

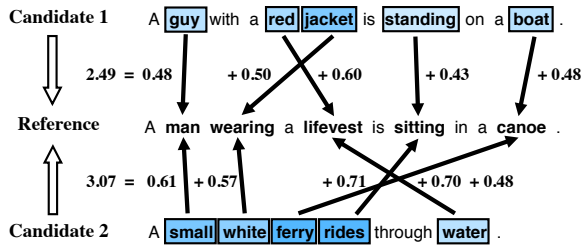


Figure 2: An illustration of the distance calculation of WMD metric comparing two candidate captions with a reference caption.

word2vec embedding space. Specifically, the distance between word i and word j in two documents is set as the Euclidean distance between each of the corresponding *word2vec* embeddings x_i and x_j , *i.e.*, $c(i, j) = \|x_i - x_j\|_2$.

The distances between words serve as building blocks to define distances between documents, hence captions. The flow between word vectors is defined with the sparse flow matrix $\mathbf{T} \in R^{n \times n}$, with T_{ij} representing the travel amount of word i to word j . The distance between two documents is then defined with $\sum_{i,j} T_{ij} c(i, j)$, *i.e.* the minimum cumulative cost required to move all words between documents. This minimum cumulative cost is found by solving the corresponding linear optimization problem, which is cast as a special case of EMD metric (Rubner et al., 2000). An example matching result is shown in Figure 2. By using *word2vec* embeddings, semantic similarities between words are more accurately identified. In our experiments, we convert the distance scores to similarities by using a negative exponential.

2.7 Drawbacks of the metrics

In order to illustrate the drawbacks of these automatic evaluation metrics, we provide an exam-

ple case in Table 2. In this table, an original caption is given, together with the upper bound values for each metric, *i.e.* when this original caption is compared to itself. The second line includes a candidate caption that is semantically very similar to the original one and the corresponding similarity scores according to evaluation metrics. We then modify the candidate sentence slightly and observe how the metric scores are affected from these small modifications. First, we observe that all the scores decrease when some words are replaced with their synonyms. The change is especially significant for SPICE and CIDEF. In this example, failure of SPICE is likely due to incorrect parsing or the failure of synonym matching. On the other hand, failure of CIDEF is likely due to unbalanced *tf-idf* weighting. Second, we observe that the metrics are not affected much from the introduction of additional (redundant) words in the sentences. However, when the order of the words are changed, we see that BLEU, ROUGE and CIDEF scores decrease notably, due to their dependence on n -gram matching. Note that, WMD and SPICE are not influenced from the change in word order.

3 Evaluation and Discussion

3.1 Quality

A common way of assessing the performance of a new automatic image captioning metric is to analyze how well it correlates with human judgments of description quality. However, in the literature, there is no consensus on which correlation coefficient is best suited for measuring the soundness of a metric in this way. Elliott and Keller (2014) reports Spearman’s rank correlation, which measures a monotonic relation, whereas Anderson *et al.* (2016) suggests to use Pearson’s correlation, which assumes that the relation is lin-

ear, and Kendall’s correlation, which is another rank correlation measure.

The above correlation analysis is a well-established practice for automatic metric evaluation, but it is not complete in the sense that it is not meaningful to draw conclusions from it about the differences or similarities between a pair of metrics. That is, comparing the corresponding correlations relative to each other does not say much since they are both computed on the same dataset, and thus not independent. To address this issue, Graham and Baldwin (2014) have suggested to use Williams significance test (Williams, 1959), which also takes into account the degree to which the two metrics correlate with each other, and can reveal whether one metric significantly outperforms the other. The test has shown to be valuable for evaluation of document and segment-level machine translation (Graham and Baldwin, 2014; Graham et al., 2015; Graham and Liu, 2016) and summarization metrics (Graham, 2015). In this study, we extend the previous correlation-based evaluations of image captioning metrics by providing a more conclusive analysis based on Williams significance test.

Williams test (Williams, 1959) calculates the statistical significance of differences in dependent correlations, and formulated as testing whether the population correlation between X_1 and X_3 equals the population correlation between X_2 and X_3 :

$$t(n-3) = \frac{(r_{13} - r_{23})\sqrt{(n-1)(1+r_{12})}}{\sqrt{2K\left(\frac{n-1}{n-3}\right) + \frac{(r_{23}+r_{13})^2}{4}(1-r_{12})^3}} \quad (1)$$

where r_{ij} is the correlation between X_i and X_j , and n is the size of the population, with

$$K = 1 - r_{12}^2 - r_{13}^2 - r_{23}^2 + 2r_{12}r_{13}r_{23}. \quad (2)$$

To analyze statistical significance in the automatic metrics listed in Section 2, we use the publicly available FLICKR-8K (Elliott and Keller, 2014) and COMPOSITE (Aditya et al., 2015) datasets, which we describe below. We note that in our experiments, we first lowercase and tokenize the candidate and reference captions using `ptbtokenizer.py` script from MS COCO evaluation tools¹. We use the implementations of the metrics from the same evaluation kit with the ex-

¹<https://github.com/peteanderson80/coco-caption>

ception of WMD. For the WMD metric, we employ the code provided by Kusner et al. (2015)².

FLICKR-8K³ dataset contains quality judgements for 5822 candidate sentences for the images in its test set (Hodosh et al., 2013). These judgements are collected from 3 human experts and they are on a scale of [1, 4], with a score of 1 denoting a description totally unrelated to the image content, and 4 meaning a perfect description for the image. Candidate captions are all obtained from a retrieval based model, hence they are grammatically correct.

COMPOSITE⁴ dataset contains human judgements for 11,985 candidate captions for the subsets of FLICKR-8K (Hodosh et al., 2013), FLICKR-30K (Young et al., 2014) and MS COCO (Lin et al., 2014) datasets. The AMT workers were asked to judge the candidate caption for an image using two aspects: (i) *correctness*, and (ii) *thoroughness* of the candidate caption, both on a scale of [1, 5] where 1 means not relevant/less detailed and 5 denotes the candidate caption perfectly describing the image. Candidate captions were sampled from the human reference captions and the captioning models in (Aditya et al., 2015; Karpathy and Fei-Fei, 2015).

Table 3 shows Pearson’s, Spearman’s and Kendall’s correlation of the metrics with the human judgements in FLICKR-8K and COMPOSITE datasets. For FLICKR-8K, we follow the methodology in (Elliott and Keller, 2014) and compute correlations with the human expert scores. On the other hand, for COMPOSITE, we report the mean of the correlations with *correctness* and *thoroughness* scores. In terms of these correlations, while SPICE produces the highest quality comparisons in FLICKR-8K, WMD and METEOR give better results in COMPOSITE in general. However, if one further inspects the score distributions of the metrics (on FLICKR-8K dataset) shown in Figure 3, while SPICE can identify irrelevant captions remarkably well, it can not effectively distinguish bad captions from relatively better ones.

In Figure 4(a), we show Spearman’s correlation between each pair of metrics, where the metrics are ordered from highest to lowest correlation with

²<https://github.com/mkusner/wmd>

³<https://github.com/elliotttd/compareImageDescriptionMeasures>

⁴<https://imagesdg.wordpress.com/image-to-scene-description-graph>

Table 3: Correlation between automatic image captioning metrics and human judgement scores.

	FLICKR-8K			COMPOSITE		
	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
WMD	0.68	0.60	0.48	0.43	0.43	0.32
SPICE	0.69	0.64	0.56	0.40	0.42	0.34
CIDEr	0.60	0.56	0.45	0.32	0.42	0.32
METEOR	0.69	0.58	0.47	0.37	0.44	0.33
BLEU	0.59	0.44	0.35	0.34	0.38	0.28
ROUGE	0.57	0.44	0.35	0.40	0.39	0.29

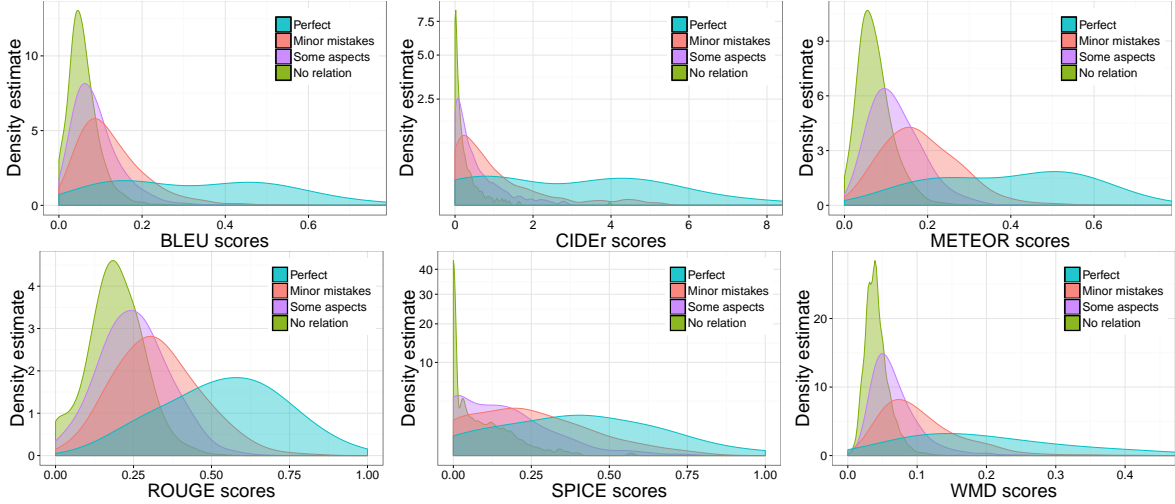


Figure 3: Score distributions of the metrics on FLICKR-8K dataset. Four different rating scales are used: 1 for no relation, 2 for minor mistakes, 3 for some true aspects and 4 for perfect match. For CIDEr and SPICE metrics, square-root transform is performed on the y -axis to better illustrate how the score distributions overlap with each other.

human judgements⁵. Overall, the pairwise correlations are generally high for both datasets. We additionally observe that the metrics which depend on similar structures are grouped together using these correlations. For example, the n -gram based metrics BLEU and ROUGE provide scores that are highly correlated with each other for FLICKR-8K. The correlations within COMPOSITE dataset are even very high for all the metrics that consider n -grams, namely BLEU, CIDEr, METEOR and ROUGE. On the other hand, the correlations of these metrics against SPICE and WMD are not that high. Moreover, the pairwise correlations between SPICE and WMD are relatively low as well. All these findings suggest that these three groups of metrics, the n -gram based metrics, the scene-graph based SPICE and the word embedding

⁵Here, we only report Spearman’s correlation since, compared to Pearson’s, it provides a more consistent ranking of the metrics across the two datasets, and is similar to Kendall’s correlation.

based WMD, can be complementary to each other.

Finally, in Figure 4(b), we provide the results of Williams significance test, which compares two different metrics with respect to their correlations against human judgements. Our results show that all the metric pairs have a significant difference in correlation with human judgement at $p < 0.05$. This reveals that the pair of metrics which has close correlation scores with human judgements (e.g. SPICE and WMD in FLICKR-8K dataset) are found to be statistically different than each other. These findings collectively support our previous conclusion that all metrics considered here can complement each other in evaluating the quality of the generated captions.

3.2 Accuracy

In this section, following the methodology introduced in (Vedantam et al., 2015), we analyze the ability of each metric to discriminate certain pair of captions from one another in reference to

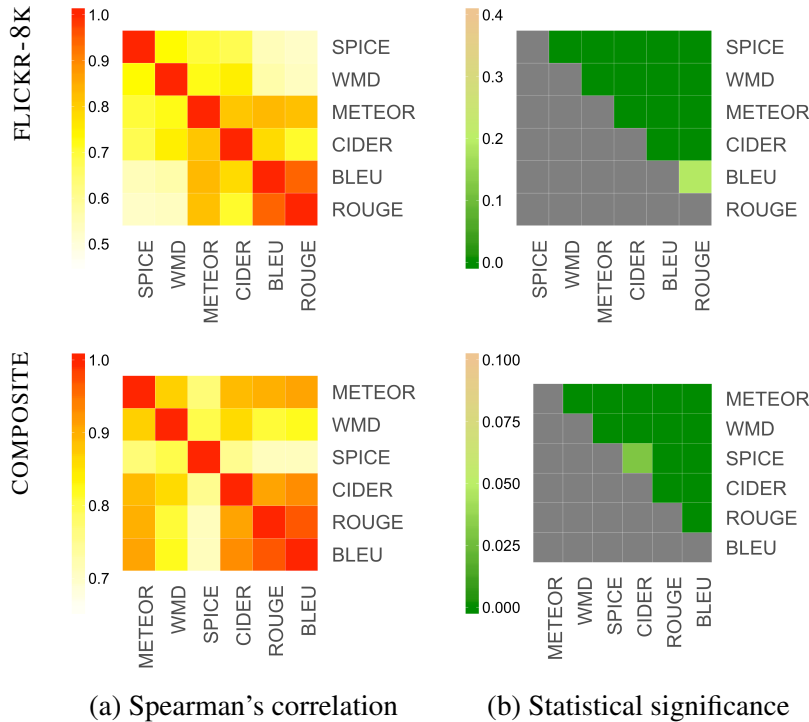


Figure 4: Significance test results for pairs of automatic metrics on FLICKR-8K and COMPOSITE datasets. (a) Spearman correlation between pairs of metrics; and (b) p -value of Williams significance tests, green cells indicate a significant win for metric in row i over metric in column j .

a groundtruth caption. We employ the human consensus scores while evaluating the accuracies. In particular, for evaluation, a triplet of descriptions, one reference and two candidate descriptions, is shown to human subjects and they are asked to determine the candidate description that is more similar to the reference. A metric is accurate if it provides a higher score to the description chosen by the human subject as being more similar to the reference caption. For this analysis, we carry out our experiments on PASCAL-50S and ABSTRACT-50S datasets⁶. We consider different kinds of pairs such as (human-human correct) HC, (human-human incorrect) HI, (human-machine) HM, and (machine-machine) MM. As the candidate sentences are generated by both humans and machines, each test scenario has a different level of difficulty.

ABSTRACT-50S (Vedantam et al., 2015) dataset is a subset of the Abstract Scenes Dataset (Zitnick and Parikh, 2013), which includes 500 images containing clipart objects in everyday scenes. Each image is annotated with 50 different descriptions. For evaluation, 48 of these 50 descriptions are used as reference descriptions and the remain-

ing 2 descriptions are employed as candidate descriptions. For 400 pairs of these descriptions, human consensus scores are available, with the first 200 are for HC and the remaining 200 are for HI.

PASCAL-50S (Vedantam et al., 2015) dataset is an extended version of the Pascal Sentences (Farhadi et al., 2010) dataset that contains 1000 images from PASCAL Object Detection challenge (Everingham et al., 2010) of 20 different object classes like person, car, horse, etc. This version includes 50 captions per image and human judgements for 4000 candidate pairs for the aforementioned binary-forced choice task, which are all collected through Amazon Mechanical Turk (AMT). For this dataset, all four different categories are available, having 1000 pairs for each category.

In Table 4, we present caption-level classification accuracy scores of automatic evaluation metrics at matching human consensus scores. On ABSTRACT-50S dataset, the CIDER metric outperforms all other metrics in both HC and HI cases. On the other hand, on PASCAL-50S dataset, the WMD metric gives the best scores in three out of four cases. Especially, it is the most accurate metric at matching human judgements on the chal-

⁶<http://vrarna91.github.io/cider>

Table 4: Description-level classification accuracies of automatic evaluation metrics.

	ABSTRACT-50S			PASCAL-50S				
	HC	HI	Avg.	HC	HI	HM	MM	Avg.
WMD	0.65	0.93	0.79	0.71	0.99	0.93	0.74	0.84
SPICE	0.62	0.89	0.76	0.66	0.98	0.85	0.72	0.81
CIDEr	0.76	0.95	0.86	0.69	0.99	0.94	0.66	0.82
METEOR	0.60	0.90	0.75	0.69	0.99	0.90	0.65	0.81
BLEU	0.69	0.89	0.79	0.67	0.97	0.94	0.60	0.80
ROUGE	0.65	0.89	0.77	0.68	0.97	0.92	0.60	0.79



Distractor Type	Gold Caption	Distracted Version
“Replace Scene”	a man wearing a life jacket is in a small boat on a lake with a ferry in view	a man wearing a life jacket is in a small boat on takeoff with a ferry in view
“Replace Person”	a man wearing a life jacket is in a small boat on a lake with a ferry in view	a woman in a blue shirt and headscarf is in a small boat on a lake with a ferry in view
“Share Person”	a man wearing a life jacket is in a small boat on a lake with a ferry in view	a man is selecting a chair from a stack under a shady awning
“Share Scene”	a man wearing a life jacket is in a small boat on a lake with a ferry in view	a black and brown dog is playing on the ice at the edge of a lake

Figure 5: Distracted versions of an image descriptions for a sample image.

Table 5: Distraction analysis.

Case	# Instances	BLEU	METEOR	ROUGE	CIDEr	SPICE	WMD
Replace-Scene	2514	0.62	0.69	0.63	0.83	0.54	0.76
Replace-Person	5817	0.73	0.77	0.78	0.78	0.67	0.80
Share-Scene	2621	0.79	0.85	0.79	0.81	0.70	0.87
Share-Person	4596	0.78	0.85	0.78	0.83	0.67	0.88
Overall	15548	0.73	0.79	0.75	0.81	0.65	0.83

lenging MM and HC cases, which require distinguishing fine-grained differences between descriptions. On average, the performances of all the other metrics are very similar to each other.

3.3 Robustness

In this section, we evaluate the robustness of the automatic image captioning metrics. For this purpose, we employ the binary (two-alternative) forced choice task introduced in (Hodosh and Hockenmaier, 2016) to compare the existing image captioning models. For a given image, this task involves distinguishing a correct description from its slightly distracted incorrect versions. In our case, a robust image captioning metric should always choose the correct caption over the distracted ones.

In our experiments, we use the data⁷ provided

⁷http://nlp.cs.illinois.edu/HockenmaierGroup/Papers/VL2016/HodoshHockenmaier16_BinaryTasks_Data.tar

by the authors for a subset of FLICKR-30K (Hodosh et al., 2013). Specifically, we consider four different types of distractions for the image descriptions, namely 1) Replace-Scene, 2) Replace-Person, 3) Share-Scene, and 4) Share-Person, which results 15548 correct and distracted caption pairs in total. For Replace-Scene and Replace-Person tasks, the distracted descriptions were artificially constructed by replacing the main actor (first person) and the scene in the original caption by random person and scene elements, respectively. For Share-Scene and Share-Person tasks, the distracted captions were selected from the sentences from the training part of FLICKR-30K (Young et al., 2014) dataset whose actor or scene chunks share the similar main actor or scene elements with the correct description. Figure 5 presents an example image together with the original description and its distracted versions.

We compare each correct caption available for an image with the remaining correct and distracted

captions for that image by considering tested evaluation metrics, and then estimate an average accuracy score. In Table 5, we present the classification accuracies of the evaluation metrics for each distraction type. As can be seen, the WMD metric gives the best results for three out of four categories, and provides the second best result for the Replace-Scene case. Overall, METEOR and CIDER metrics seem to be also robust to these distractions. The very recently proposed SPICE metric performs the worst for this task. This is somewhat expected as it is even affected by the use of synonyms of the words as we have previously shown in Table 2.

3.4 Discussion

As the experiments on quality, accuracy and robustness tests demonstrate in Sections 3.1-3.3, existing automatic image captioning metrics all have some strengths and weaknesses due to their design choices. For example, while SPICE, METEOR and WMD give the best performances in terms of our correlation analysis against human judgments, CIDER and WMD provide the best classification scores for our accuracy experiments. Moreover, CIDER, METEOR and WMD are found to be less affected by the distractors. Overall, our analysis suggests that the recently proposed WMD document metric is also quite effective for image captioning since it has high correlations with the human scores, is much less sensitive to synonym swapping and additionally performs well at the accuracy and distraction tasks.

Our analysis also shows that the existing metrics both theoretically and empirically differ from each other with significant differences. Compared to the recent results of significance testing of machine translation and summarization metrics (Graham and Baldwin, 2014; Graham et al., 2015; Graham and Liu, 2016; Graham, 2015), our results suggest that there remains much room for improvement in developing more effective image captioning evaluation metrics. We leave this for future work, but a very naive idea would be combining different metrics into a unified metric and we simply test this idea using score combination, after normalizing the score of each metric to the range $[0, 1]$. Among all possible combinations, we find that the combination of WMD+SPICE+METEOR performs the best with a Spearman’s correlation of 0.66 for FLICKR-8K and

0.45 for COMPOSITE dataset, yielding an improvement from SPICE (0.64 and 0.42). In addition, we should add that this unified metric significantly outperforms the individual metrics according to Williams test ($p < 0.01$).

4 Conclusion

In this paper, we provide a careful evaluation of the automatic image captioning metrics, and propose to use WMD, which utilizes *word2vec* embeddings of the words to compute a semantic similarity of sentences. We highlight the drawbacks of the existing metrics, and we empirically show that they are significantly different than each other. We hope that this work motivates further research into developing better evaluation metrics, probably learning based ones, as previously studied in machine translation literature (Kotani and Yoshimi, 2010; Guzmán et al., 2015). We also observe that incorporating visual information (via Scene-graph used by SPICE) and semantic information (via WMD) is useful for the caption evaluation task, which motivates the use of multimodal embeddings (Kottur et al., 2015).

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK), with award no 113E116.

References

- Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos. 2015. From images to sentences through scene description graphs using commonsense reasoning and knowledge. *arXiv preprint arXiv:1511.03292*.
- Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic propositional image caption evaluation. In *ECCV*.
- Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.
- Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank.

2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *JAIR*, 55:409–442.
- Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *EMNLP*, volume 13, pages 1292–1302.
- Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *ACL*, pages 452–457.
- Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The Pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338.
- Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*, pages 15–29.
- Yvette Graham and Timothy Baldwin. 2014. Testing for significance of increased correlation with human judgment. In *EMNLP*, pages 172–176.
- Yvette Graham and Qun Liu. 2016. Achieving accurate conclusions in evaluation of automatic machine translation metrics. In *NAACL-HLT*.
- Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2015. Accurate evaluation of segment-level machine translation metrics. In *NAACL-HLT*.
- Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *EMNLP*, pages 128–137.
- Francisco Guzmán, Shafiq Joty, Lluís Márquez, and Preslav Nakov. 2015. Pairwise neural machine translation evaluation. In *ACL-IJNLP*, pages 805–814.
- Micah Hodosh and Julia Hockenmaier. 2016. Focused evaluation for image description with binary forced-choice tasks. In *VL*.
- Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*, 47:853–899.
- Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. 2015. Guiding long-short term memory for image caption generation. *arXiv preprint arXiv:1509.04942*.
- Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A Shamma, Michael S Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *CVPR*, pages 3668–3678.
- Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137.
- Katsunori Kotani and Takehiko Yoshimi. 2010. A machine learning-based evaluation method for machine translation. In *SETN*, pages 351–356.
- Satwik Kottur, Ramakrishna Vedantam, José MF Moura, and Devi Parikh. 2015. Visual Word2Vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. *arXiv preprint arXiv:1511.07067*.
- Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2013. Babytalk: Understanding and generating simple image descriptions. *IEEE Trans. Pattern Anal. Mach. Intel.*, 35(12):2891–2903.
- Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *ICML*, pages 957–966.
- Siming Li, Girish Kulkarni, Tamara L. Berg, Alexander C. Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In *CoNLL*, pages 220–228.
- Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL*, pages 605–612.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*, pages 740–755.
- Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.
- T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.
- Ted Pedersen, Siddharth Patwardhan, and Jason Michellizzi. 2004. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41.
- Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The earth mover’s distance as a metric for image retrieval. *IJCV*, 40(2):99–121.
- Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *VL*, pages 70–80.

- Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.
- Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.
- Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164.
- Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2016. Show and Tell: Lessons learned from the 2015 MSCOCO image captioning challenge. *arXiv preprint arXiv:1609.06647*.
- Evan J. Williams. 1959. *Regression analysis*, volume 14. Wiley New York.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.
- Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78.
- C Lawrence Zitnick and Devi Parikh. 2013. Bringing semantics into focus using visual abstraction. In *CVPR*, pages 3009–3016.

Integrating Meaning into Quality Evaluation of Machine Translation

Osman Başkaya¹, Eray Yıldız¹, Doruk Tunaoglu¹, M. Tolga Eren¹, and A. Seza Doğruöz²

¹Huawei Turkey Research and Development Center, Istanbul, Turkey
{osbaskaya, doruktuna, tolgaeren}@gmail.com, eray.yildiz@huawei.com

²Independent Researcher
a.s.dogruoz@gmail.com

Abstract

Machine translation (MT) quality is evaluated through comparisons between MT outputs and the human translations (HT). Traditionally, this evaluation relies on form related features (e.g. lexicon and syntax) and ignores the transfer of meaning reflected in HT outputs. Instead, we evaluate the quality of MT outputs through meaning related features (e.g. polarity, subjectivity) with two experiments. In the first experiment, the meaning related features are compared to human rankings individually. In the second experiment, combinations of meaning related features and other quality metrics are utilized to predict the same human rankings. The results of our experiments confirm the benefit of these features in predicting human evaluation of translation quality in addition to traditional metrics which focus mainly on form.

1 Introduction

Machine translation (MT) systems translate large chunks of data automatically across languages. Although these systems may achieve high level accuracies using form related features (e.g. lexical and syntactic), they often fail to carry over the meaning embracing the form. Example (1) highlights the meaning difference between a Human Translation (HT) and an MT output for the same source sentence:

Example (1)

HT: “*Your feet’s too big.*”¹

MT: “*Your feet is too great.*”²

Although the form is often preserved, MT outputs may sound “strange” or “different” in comparison to HT ones due to the loss of meaning. Therefore, human translators generally enrich the text with the appropriate tone, style and sentiments during translation. Current quality evaluation metrics like BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007) are based on form related features and do not directly consider the transfer of meaning (e.g. sentiment and style) in MT. Some of these metrics check for synonyms and paraphrases but this approach is still limited to the coverage of the corresponding pair tables. In other words, these metrics do not explicitly evaluate the transfer of meaning in MT. Our main goals are:

- to find out whether the transfer of meaning related features (e.g. sentiment and style) in MT influences the human judgment of translation quality.
- to compare meaning and form related features for quality evaluation of MT.
- to measure whether meaning and form related features can be combined to improve the performance of existing MT quality evaluation metrics.

¹WMT’15 Finnish to English test set, reference translation, segment id:440

²WMT’15 Finnish to English test set, translated by system: UoS.4059, segment id:440

By using publicly available parallel corpora (Tenth Workshop on Statistical Machine Translation (WMT15)), we achieve our goals with two experiments described in Section 5. Our results indicate that combining meaning related features with form related ones approximates to the human judged rankings better than the BLEU metric. These combined features also improve the performance of other MT quality evaluation metrics by 0.5-2 percentage points.

2 Related Work

So far, MT studies have focused mostly on features related to form (e.g., lexical and syntactic features) for the automatic evaluation of MT quality (e.g., BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007)). BLEU metric is based on n-gram matching of the HT and MT texts and used widely in the MT community to evaluate the MT quality. METEOR employs both word matching scores and the linguistic information (e.g., synonyms and stemming) in contrast to BLEU. Following studies have evaluated MT quality with various features: POS tags (Dahlmeier et al., 2011), morphemes (Tantuĝ et al., 2008), sentence structure (Li et al., 2012), named entities (Buck, 2012), semantic textual similarity (Castillo and Estrella, 2012), paraphrasing (Snover et al., 2006), semantic roles (Lo and Wu, 2011) and language models (Stanojevic and Simaan, 2014). Recently, Yu et al. (2015) proposed another metric (i.e. DPMFComb) which is a combination of a syntax-based metric and some other evaluation metrics in Asiya³. At WMT15, DPMFComb obtained the best results at the metrics task for system-level evaluation of translation into English tasks.

Although previous methods require human reference translation, recent methods (e.g. *quality estimation metrics*), aim to eliminate the necessity of human translation. These methods apply Machine Learning (ML) techniques using lexical (e.g. average source/target token length), syntactic (e.g. ratio of percentage of POS tags in the source/target sentences), and statistical features (e.g. source/target sentence LM probability, word alignment probabilities, etc.) (Stymne et al., 2014; Langlois, 2015; Shah et al., 2015). Interested reader may also benefit from the survey on MT evaluation metrics by Han and Wong (2016).

³<http://asiya-faust.cs.upc.edu/>

Src. Lng.	Domain	# of Sent's	# of Jdg's
Czech	News Texts	2496	20224
Finnish	News Texts	1744	10757
French	News Forum	2136	12189
German	News Texts	1989	12880
Russian	News Texts	2407	14924
Total		10772	70974

Table 1: WMT15 Test Data Statistics grouped by source languages. The domain of source text, the number of sentences and the number of human judgments are presented for each source language.

Chen and Zhu (2014) explore sentiment consistency between MT and HT texts to improve the MT quality by incorporating sentiment related features (e.g. subjectivity, polarity, intensity and negation). By using these features in their MT system, they improved the BLEU score by 1.1 point on NIST Chinese-to-English translation dataset⁴. Mohammad et al. (2015) also investigate the sentiment consistency between MT and HT texts with a different motivation. They improve sentiment analysis performance for Arabic by translating available resources (e.g., sentiment lexicon, sentiment annotated data) from English to Arabic. Although sentiment analysis of English translations of Arabic texts obtain competitive results to current state-of-the-art Arabic sentiment analysis systems, they did not evaluate the MT output quality.

There are also studies using MT systems to enrich labeled data for sentiment analysis by translating between languages and leveraging sentiment scores (Wan, 2009; Demirtaş and Pechenizkiy, 2013; Hiroshi et al., 2004). However, none of these studies employ meaning related features to evaluate the MT quality.

3 Data Description

3.1 2015 Workshop on Statistical Machine Translation

We utilized WMT15⁵ parallel corpora (Bojar et al., 2015) which include several tasks (e.g., standard news translation task, a metrics task, a tuning task, a task for run-time estimation of machine translation quality, and an automatic post-editing task). 24 institutions participated in the translation task with a total of 68 machine translation systems. The WMT15 data includes:

⁴<http://www.nist.gov/speech/tests/mt>

⁵<http://www.statmt.org/wmt15/>

	Human Translation (HT) Output	Machine Translation (MT) Output
1.	<i>Adam, you see badly what you are looking at.</i>	<i>Adam, you see what you look at.</i>
2.	<i>Of course I don't hate you.</i>	<i>Of course I hate you.</i>
3.	<i>This is business news</i>	<i>This is supposed to be of business news</i>
4.	<i>The views of Chinese towards white people is similar!</i>	<i>The Chinese think like white people!</i>

Table 2: Examples of MT Errors in WMT15 Dataset

- Source sentences
- Reference human translations (HT)
- Machine translations (MT)
- Human judgments (e.g. from 1 (*best*) to 5 (*worst*)) for each MT text.

The data is available for five language pairs: Czech (ces)-English, French (fre)-English, German (deu)-English, Finnish (fin)-English, and Russian (rus)-English. Domains of the test data are the same for all languages except for French. The test data for the French-English language pair was fetched from a news discussion forum instead of news texts. Table 1 shows the statistics for the test data. The target language is English for all source languages. The domain of source text, the number of sentences and the number of human judgments are presented. All data was based on the news text corpora except for French-English pair.

In order to evaluate the quality of each MT system, Bojar et al. (2015) conducted a human evaluation using Appraise⁶ (Federmann, 2012) which is an open source toolkit (similar to Amazon Mechanical Turk⁷). Each segment consists of a source sentence in the original language (e.g. Czech), its corresponding human translation (English), and 5 anonymous MT system translations (English).

To make the task more consistent and to increase the number of data points, the organizers treated almost identical system translations as one. Even though exactly 5 translations are presented to each judge in a segment, there may be more than 5 MT systems that are ranked. Judges rate the segments from 1 (best) to 5 (worst) by the quality of translated sentences (allowing ties).

⁶<https://github.com/cfedermann/Appraise>

⁷<https://www.mturk.com/mturk/>

In total, there were 29,007 segments, each of which would have produced at least 10 individual system comparisons (e.g., $A > B$, $B > C$, $A = C$, $C > B$, etc.). To map these individual comparisons to system scores, the organizers used TrueSkill⁸ (Herbrich et al., 2006), a Bayesian skill ranking algorithm (similar to Elo used in Chess (Elo, 1978)) and fed these individual bilateral comparisons to TrueSkill. A score is produced for each participated system. In this study, we utilized the HT texts, MT system translations and human judgments in our experiments.

3.2 Features

Table 2 provides examples of MT errors in comparison to HT. All example translations (MT vs. HT texts) are selected from the WMT15 dataset based on the lowest (5) rankings by human judges. Although translations overlap at the word level, they convey quite different meanings. In example (1), the word '*badly*' has disappeared in MT output and led to a loss of information. In example (2), a negated sentence is translated as an affirmative sentence by the MT system. Example (3) illustrates how the MT system generates a more speculative sentence than HT sentence. The pair in example (4) differs in terms of formality between MT vs HT output. MT evaluation metrics may attribute high scores for these pairs since they mainly focus on lexical and syntactic matching. However, as our examples demonstrate, meaning could easily be lost if we rely only on form related MT system evaluation metrics.

To investigate the consistency between MT and HT texts for sentiment and stylistic features, we make use of *sentiment polarity*, *subjectivity*, *connotation*, *negation*, *speculation*, *readability* and *formality* to measure how these features influence the quality of translation with respect to human rankings.

⁸<http://research.microsoft.com/en-us/projects/trueskill/>

Sentiment Polarity indicates whether the designated sentence has an affirmative or negative sentiment. To measure the impact of this feature, we use *Vader*, a rule based sentiment analysis tool (Hutto and Gilbert, 2014). It utilizes grammatical and syntactical rules. In the experiments performed by Hutto and Gilbert (2014), *Vader* outperforms several competing sentiment analysis approaches.

Additionally, we trained a machine learning (ML) based sentiment analyzer using a deep learning approach described by Yildiz et al. (2016). Their architecture is a Convolutional Neural Network (CNN) which takes pre-trained word vectors⁹ as input and applies interleaved convolution and pooling operations. The top layer in the network is Softmax layer which computes the probability of assigning a class (positive, negative).

We adopted this architecture and trained a network using Stanford Twitter Sentiment Corpus¹⁰. The training set contains 1.6 million tweets automatically labeled as positive or negative from various domains while the test set is labeled manually. This ML based sentiment analyzer achieves 90.1% accuracy and outperforms the SVM classifier reported by Go et al. (2009).

Subjectivity indicates whether a text expresses an opinion. In order to compute the subjectivity scores, we trained our architecture using the *sentiment polarity and subjectivity dataset*¹¹ (Pang and Lee, 2004) which includes 5000 subjective and 5000 objective sentences. We applied 10-fold cross validation to the data and obtained 91.50% average accuracy.

Connotation indicates cultural or emotional association carried by words that appear in sentences (Feng et al., 2013). In contrast to the sentiment polarity, connotation polarity indicates subtle shades of sentiment beyond denotative or surface meaning of text. The words which do not express sentiment can carry a positive or negative connotation.

For instance, “life” and “home” are considered neutral with regard to the sentiment analysis. However, they convey a positive connotation

(Carpuat, 2015). We use the connotation polarity of each word in a sentence to compute connotation score using a normalized version of the formulation proposed by Carpuat (2015). The connotation polarities of the words are obtained by looking up a lexicon which is constructed by Feng et al. (2013). We used the following formula to compute the connotation score:

$$CS = \frac{\#positive - \#negative}{\#total} \quad (1)$$

where CS is the connotation score, $\#positive$ indicates the number of the words with positive connotation and $\#negative$ indicates the number of the words with negative connotation.

This formula assigns a continuous value between 1 and -1 to the sentence as a connotation score. The values close to 1 indicate that a given sentence carries a positive connotation while the values close to -1 indicate a negative connotation.

Negation turns an affirmative statement into a negative one. We also detect the effects of negation feature in our experiments. Konstantinova et al. (2012) present a freely available dataset which contains 400 reviews (50 each from 8 domains such as movies and consumer products) annotated by linguists for negation and speculation. We train our deep learning model with these datasets and obtain 96.65% accuracy for negation.

Speculation is used to express levels of certainty. We obtain 95.55% accuracy using the same dataset and method for negation.

Readability measures the ease of reading and comprehending a text (Dale and Chall, 1948). For readability measurement we use Flesch reading-ease test in which higher scores indicate that the text is easier to read. The Flesch readability score (Kincaid et al., 1975) is calculated using the sentence length and the number of syllables per word as presented in the formula below.

$$Flesch = 206.835 - 1.015 \frac{A}{B} - 84.6 \frac{C}{A} \quad (2)$$

where A is the number of words, B is the number of sentences and C is the number of syllables in a given text.

In addition to the rule based readability measurement, we use an ML based readability metric “*simplicity*” as described by Vajjala and Meurers (2016). They extract various syntactic, psycholinguistic and lexical features from text and

⁹<https://code.google.com/archive/p/word2vec/>

¹⁰<http://help.sentiment140.com/for-students>

¹¹http://www.cs.cornell.edu/people/pabo/movie-review-data/rotten_imdb.tar.gz

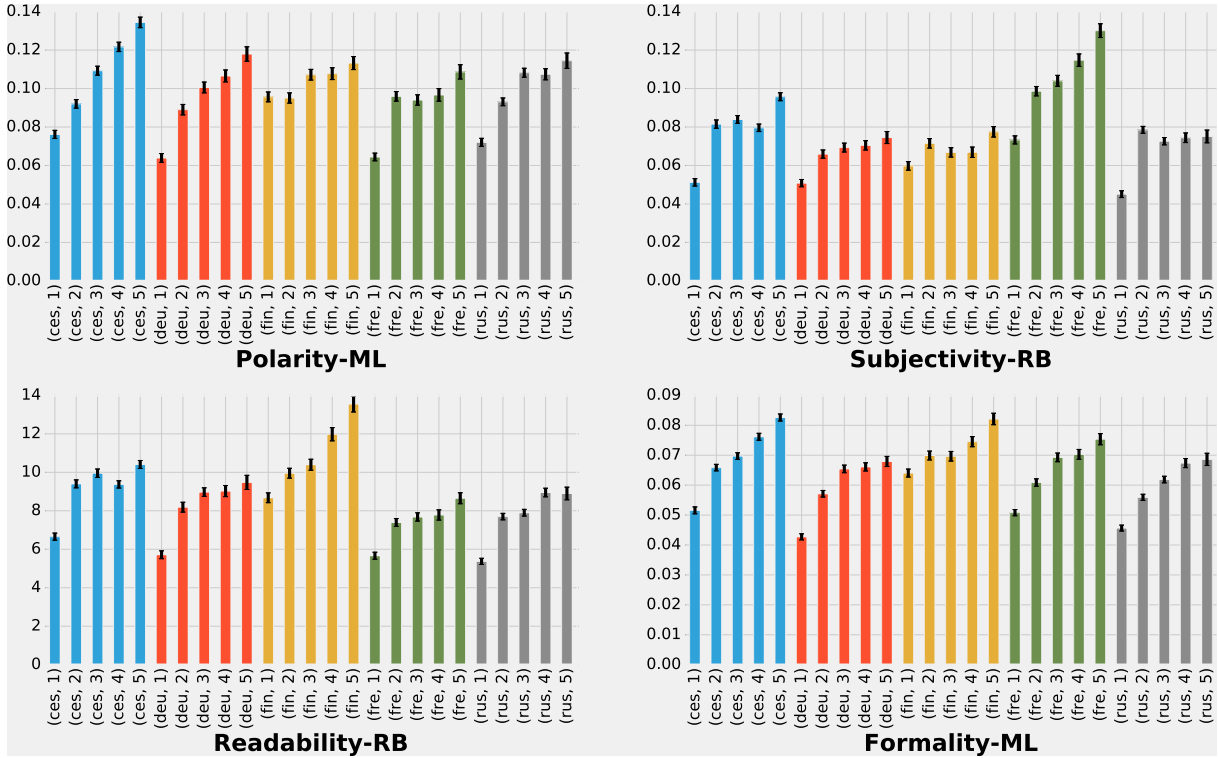


Figure 1: Means of absolute differences between the feature scores of MT and HT outputs. x-axis denotes the language and human rank pairs. Error bars indicate 95% confidence intervals.

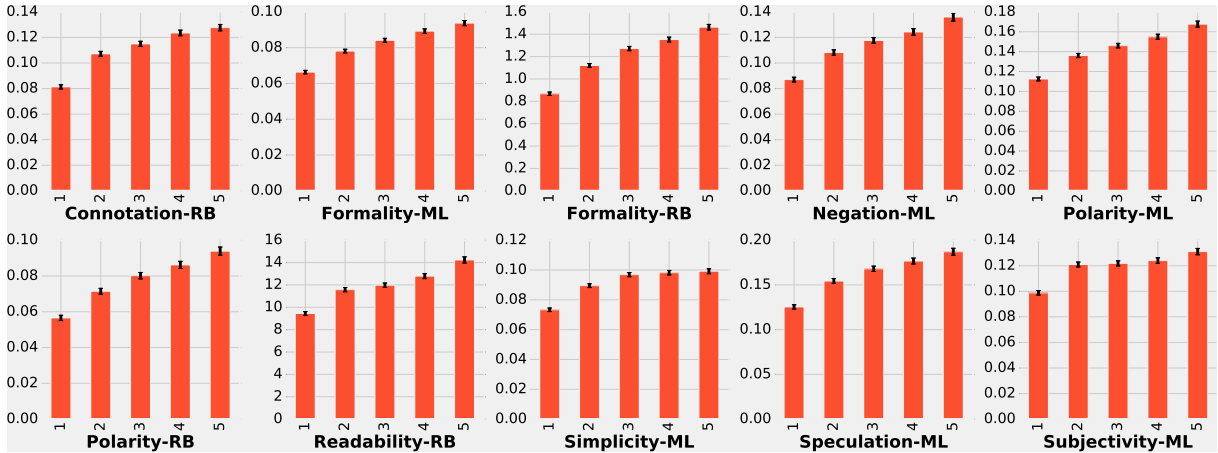


Figure 2: Means of absolute differences between the features scores of MT outputs and the corresponding HT outputs for all features. x-axis denotes the human rankings. Error bars indicate 95% confidence intervals.

train a pair-wise classifier using them. The training data is a sentence-aligned corpus constructed from news articles and Wikipedia pages and their simplified versions. The method correctly classifies the simplified and complex sentences in terms of their reading level with an accuracy of over 80%.

Formality Heylighen and Dewaele (1999) state formality as the most important dimension of vari-

ation between styles. They define the formality score as a function of POS tag frequencies. The formality score is given in Equation 3 where NF is the frequency of nouns, AdjF is the adjective frequency, PF is the preposition frequency, ArtF is the article frequency, PrpF is the proper noun frequency, VF is the verb frequency, AdvF is the adverb frequency and IF is the interjection frequency.

$$F = \frac{NF + AdjF + PF + ArtF}{2} - \frac{PrpF + VF + AdvF + IF}{2} + 50 \quad (3)$$

Additionally, we use an ML based formality score obtained by training the mentioned architecture on the dataset introduced by Pavlick and Tetreault (2016). We have observed 80.71% accuracy through 10-fold cross validation.

All metrics were normalized between (0, 1) except the Readability and Formality. Since these two metrics are formula-based, we avoided interfering with their original scales.

4 Method

In the WMT15 task, the language pairs are divided into two groups depending on whether English is the source or the target language. We only utilize the pairs where English is the target language due to the richness in resources. For each feature, MT texts are ranked using the following approach:

1. Compute the score for HT text (e.g., 0.65).
2. Compute the scores for MT texts (e.g. A=.79, B=.25, C=.20, D=.95, E=.30).
3. Compute the absolute difference between MT scores and the HT score (e.g., $\hat{A} = .14$, $\hat{B} = .40$, $\hat{C} = .45$, $\hat{D} = .30$, $\hat{E} = .35$).
4. Rank the systems according to these differences where a smaller value corresponds to a better ranking (e.g. 1=A, 2=D, 3=E, 4=B, 5=C).

Figure 1 shows absolute differences for four features with respect to language and human rankings of MT system output. For instance, *Subjectivity-RB* feature captures the differences between ranks when the source language is French but cannot achieve the same performance for Finnish and Russian translations. Moreover, *Readability-RuleBased (RB)* and *Formality-MachineLearning (ML)* seem to perform well for all languages whereas *Polarity-ML* falls short for French.

Figure 2 illustrates the trend for all features in which absolute score difference between MT system outputs and HT text is low for high rankings (e.g., 1) and high for the low (e.g., 5) ones. Therefore, high ranked translations preserve the meaning better than the low ranked ones. Note that both

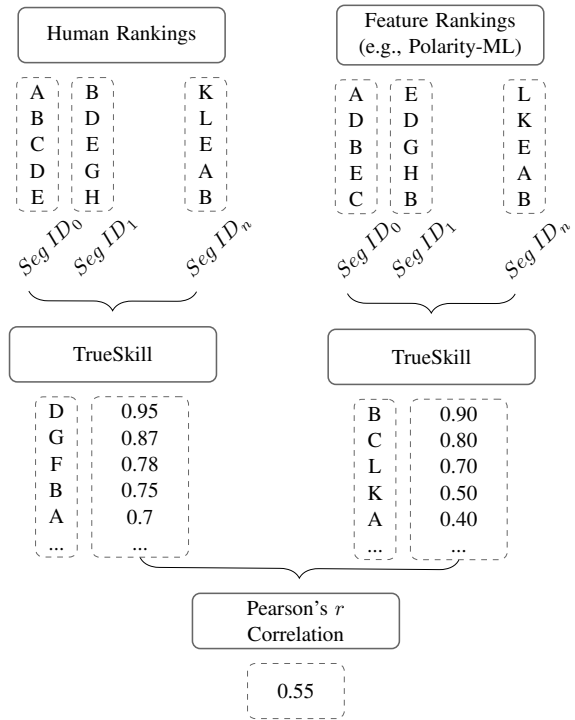


Figure 3: Steps to map human rankings and feature rankings (for example, *Polarity-ML*) to system-wide scores. *A* to *L* denotes individual rankings and *SegID_i* denotes the *i*th segment in the WMT15 test set.

figures are descriptive and do not correspond to an objective evaluation directly.

5 Experiments

5.1 Experiment #1: Impact of Individual Features on Translation Quality

This experiment investigates the correlation between each feature and MT translation quality evaluated by rankings of human judges. Using the rankings described in Section 4, we followed the “System-Based Evaluation Methodology” by Stanojević et al. (2015). After obtaining the rankings for each feature as described in the previous section, we used TrueSkill to map segment rankings to system-wide scores (see Figure 3). Next, we compared TrueSkill scores obtained per feature and human judgments with Pearson’s r correlation using the scripts provided by the WMT15 Metrics Task¹². As stated in (Stanojević et al., 2015) the script performs bootstrap resampling of 1000 samples while calculating the correlation scores and the 95% confidence intervals.

¹²<http://www.statmt.org/wmt15/metrics-task/wmt15-metrics-results.tgz>

	All	Ces	Deu	Fin	Fre	Rus
Connotation-RB	74.2 ± 2.2	87.6 ± 0.8	86.1 ± 2.0	41.7 ± 3.5	87.8 ± 1.8	67.9 ± 2.6
Formality-ML	74.9 ± 2.1	62.7 ± 1.1	85.3 ± 1.9	85.9 ± 2.0	80.3 ± 2.2	60.3 ± 3.1
Formality-RB	80.7 ± 1.7	67.9 ± 1.1	92.5 ± 1.5	68.0 ± 2.8	97.4 ± 0.9	78.0 ± 2.4
Negation-ML	48.8 ± 2.7	42.2 ± 1.4	61.2 ± 3.0	33.4 ± 3.5	78.5 ± 2.1	28.8 ± 3.6
Polarity-ML	67.1 ± 2.3	54.0 ± 1.2	78.2 ± 2.2	65.6 ± 2.7	76.1 ± 2.4	61.7 ± 2.8
Polarity-RB	78.6 ± 2.1	79.2 ± 1.0	88.6 ± 1.7	75.6 ± 2.5	81.4 ± 2.4	67.9 ± 2.7
Readability-RB	76.7 ± 2.0	66.4 ± 1.2	79.8 ± 2.2	79.7 ± 2.1	85.3 ± 2.0	72.4 ± 2.6
Simplicity-ML	41.5 ± 2.8	17.6 ± 1.4	54.9 ± 3.0	18.4 ± 3.7	77.5 ± 2.5	39.1 ± 3.5
Speculation-ML	62.2 ± 2.4	41.4 ± 1.3	68.3 ± 2.7	63.6 ± 2.9	86.2 ± 2.1	51.7 ± 3.2
Subjectivity-ML	61.1 ± 2.6	56.3 ± 1.3	66.4 ± 2.9	42.5 ± 3.2	75.9 ± 2.6	64.3 ± 2.9
BLEU	91.6 ± 1.4	95.8 ± 0.6	86.5 ± 2.0	92.9 ± 1.4	97.5 ± 0.9	85.1 ± 2.2
DPMFComb	96.2 ± 0.9	96.0 ± 0.5	97.0 ± 0.9	95.1 ± 1.2	98.0 ± 0.8	95.0 ± 1.1
Meteor	94.9 ± 1.0	94.8 ± 0.5	95.5 ± 1.0	96.3 ± 1.0	95.1 ± 1.2	92.7 ± 1.4
Random-Baseline	0.0 ± 2.9	-28.4 ± 1.5	47.6 ± 3.2	-65.9 ± 2.8	-3.6 ± 3.6	50.4 ± 3.4

Table 3: Pearson’s r correlation between Trueskill scores of a metric and human judgments with the corresponding 95% confidence intervals are shown. Each row represent either a meaning related feature (top) or a selected metric from WMT15 (bottom). ML stands for machine learning and RB stands for rule based method.

We have used three metrics from WMT15 Metrics Task for comparison, BLEU and METEOR and DPMFComb. DPMFComb was selected since it was the best system in overall score in system-based evaluation of WMT15 Metrics Shared Task and the best performing evaluation metric for three out of five languages.

Results Table 3 shows all the Pearson’s correlations. Overall, *Formality-RB* obtains the highest correlation score (80.7%) among all features. However, DPMFComb (96.2%), BLEU and METEOR are better than the rest. The exceptions are German for BLEU and French for METEOR. For German, *Formality-RB* (92.5%) outperforms BLEU (86.5%). For French, *Formality-RB* (97.4%) beats the METEOR score (95.1%). In addition, Rule-Based (RB) systems perform better than Machine Learning (ML) ones. For example, *Formality-RB* and *Polarity-RB* outperform *Formality-ML* and *Polarity-ML* respectively.

Meaning related features outperform *Random* baseline as expected. The random baseline is computed by assigning random ranks (1-5) to each translation in each segment. We assigned uniformly random ranks to all sentences without considering the language. Although its performance may vary per language, its overall performance is 0.0 (± 2.9).

5.2 Experiment #2: Impact of Combined Features on Translation Quality

As discussed in Section 4, our approach is fundamentally different than MT evaluation metrics such as BLEU. Results of our first experiment indicated strong correlations between quality scores of the features and human rankings. Therefore, we also investigate whether we can predict human rankings of MT translated text by combining these features since they capture different aspects of translation.

In contrast with Experiment 1, this experiment focuses on training systems that combine several features to predict human rankings. As input, BLEU, METEOR and DPMFComb metrics are utilized in combination with the feature scores. We experimented with several classifiers from RankLib¹³ to train the ensemble systems and opted to utilize a Random Forest (Liaw and Wiener, 2002) based approach which produced the best 5-fold cross validation score.

First, we obtained scores and rankings for each translation using the Random Forest classifiers for the following combinations:

1. All meaning related features
2. All meaning related features + BLEU

¹³<https://sourceforge.net/p/lemur/wiki/RankLib/>

	All	Ces	Deu	Fin	Fre	Rus
ALL+DPMFComb	96.8 ± 0.8	95.9 ± 0.4	97.5 ± 0.8	97.9 ± 0.8	98.6 ± 0.6	94.3 ± 1.3
DPMFComb	96.2 ± 0.9	96.0 ± 0.5	97.0 ± 0.9	95.1 ± 1.2	98.0 ± 0.8	95.0 ± 1.1
ALL+Meteor	95.8 ± 0.9	95.4 ± 0.5	96.6 ± 1.0	97.7 ± 0.8	98.0 ± 0.8	91.2 ± 1.6
Meteor	94.9 ± 1.0	94.8 ± 0.5	95.5 ± 1.0	96.3 ± 1.0	95.1 ± 1.2	92.7 ± 1.4
ALL+BLEU	93.5 ± 1.2	93.4 ± 0.6	92.3 ± 1.5	96.8 ± 0.9	97.6 ± 0.9	87.3 ± 1.9
ALL	92.0 ± 1.2	87.5 ± 0.7	93.4 ± 1.3	94.5 ± 1.2	97.8 ± 0.8	86.8 ± 1.8
BLEU	91.6 ± 1.4	95.8 ± 0.6	86.5 ± 2.0	92.9 ± 1.4	97.5 ± 0.9	85.1 ± 2.2

Table 4: Pearson’s r correlation between Trueskill scores of a metric and human judgments with the corresponding 95% confidence intervals are shown. ALL represents the combination of all meaning related features.

3. All meaning related features + Meteor
4. All meaning related features + DPMFComb

evaluation of MT system quality can be predicted with a higher accuracy. (See Table 4).

Then, we calculated the Trueskill scores for translation systems and finally fed them into WMT15 scripts to obtain Pearson’s r correlation similar to the first experiment.

Results Combined meaning related features outperform the BLEU score (Table 4). Even though the margin is relatively low, it is a promising indication. Moreover, combining them with a metric increases the performance of the metric: 1.9pp for BLEU, 0.9pp for METEOR and 0.6pp for DPMFComb. In other words, these features can utilize some meaning or style related information which is not captured by the conventional MT evaluation metrics.

6 Discussion & Conclusion

In this paper, we investigate how meaning related features influence the automatic evaluation of MT systems. Our experiments prove the additional benefit of these features in predicting human evaluation of translation quality. More specifically, we find that:

- MT systems that are ranked higher by human judges preserve the meaning (features such as polarity, formality and readability) better than the low ranked ones.
- Rankings of MT output generated according to meaning based features correlate highly with human rankings on translation quality (See Figure 2).
- When meaning related features are combined with form related lexical features, human

Extracting meaning related features from text and using form related features for MT evaluation have been studied separately. However, integrating meaning related features into MT quality evaluation can capture the meaning preservation from source to target languages. Our experiments prove that this integrated approach achieves a only slightly better performance than the form based metrics (e.g. BLEU). Moreover, our experiments indicate that the meaning related features can boost the performance of BLEU, METEOR and DPMFComb metrics without even specific optimization. Therefore, our method of integrating meaning related features to MT systems with ranking components can also improve the performances of other metrics instead of only relying on form based features.

Commonly used evaluation metrics (e.g. BLEU and METEOR) require a reference human translation to assess the quality of MT. We also use human translation as a reference since most meaning related feature extraction tools are only available for English and limited for other languages. Although there are studies assessing the quality of MT systems without human translation, meaning related features are still not integrated to MT systems yet. As new tools for other languages become available, we plan to extend our work to implement MT quality estimation for these languages as well. As future work, we will investigate the ways to develop more "human-like" MT systems by employing these meaning related and stylistic features in the training of MT systems or in post-processing steps such as parameter tuning.

Acknowledgments

This work is partially funded by 3140951 numbered TUBITAK-TEYDEB grant.

References

- Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.
- Christian Buck. 2012. Black box features for the wmt 2012 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 91–95. Association for Computational Linguistics.
- Marine Carpuat. 2015. Connotation in translation. In *6th Workshop On Computational Approaches to Subjectivity, Sentiment and Social Media Analysis WASSA 2015*, page 9.
- Julio Castillo and Paula Estrella. 2012. Semantic textual similarity for mt evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 52–58. Association for Computational Linguistics.
- Boxing Chen and Xiaodan Zhu. 2014. Bilingual sentiment consistency for statistical machine translation. In *EACL*, pages 607–615.
- Daniel Dahlmeier, Chang Liu, and Hwee Tou Ng. 2011. Tesla at wmt 2011: Translation evaluation and tunable metric. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 78–84. Association for Computational Linguistics.
- Edgar Dale and Jeanne S. Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.
- Erkin Demirtaş and Mykola Pechenizkiy. 2013. Cross-lingual polarity detection with machine translation. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 9. ACM.
- Arpad E. Elo. 1978. *The rating of chessplayers, past and present*. Arco Pub.
- Christian Federmann. 2012. Appraise: an open-source toolkit for manual evaluation of mt output. *The Prague Bulletin of Mathematical Linguistics*, 98:25–35.
- Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *ACL*, pages 1774–1784.
- Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.
- Aaron Li-Feng Han and Derek Fai Wong. 2016. Machine translation evaluation: A survey. *arXiv preprint arXiv:1605.04515*.
- Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. Trueskill: A bayesian skill rating system. In *Advances in Neural Information Processing Systems*, pages 569–576.
- Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Internet Bericht, Center Leo Apostel, Vrije Universiteit Brussel*.
- Kanayama Hiroshi, Nasukawa Tetsuya, and Watanabe Hideo. 2004. Deeper sentiment analysis using machine translation technology. In *Proceedings of the 20th international conference on Computational Linguistics*, page 494. Association for Computational Linguistics.
- Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media*.
- J. Peter Kincaid, Robert P. Fishburne Jr, Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.
- Natalia Konstantinova, Sheila CM De Sousa, Noa P. Cruz Díaz, Manuel J. Maña López, Maite Taboada, and Ruslan Mitkov. 2012. A review corpus annotated for negation, speculation and their scope. In *LREC*, pages 3190–3195.
- David Langlois. 2015. Loria system for the wmt15 quality estimation shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 323–329, Lisbon, Portugal, September. Association for Computational Linguistics.
- Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.
- Liangyou Li, Zhengxian Gong, and Guodong Zhou. 2012. Phrase-based evaluation for machine translation. In *Proceedings of COLING 2012: Posters*, pages 663–672, Mumbai, India, December. The COLING 2012 Organizing Committee.

- Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomforest. *R News*, 2(3):18–22.
- Chi-kiu Lo and Dekai Wu. 2011. Meant: an inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility via semantic frames. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 220–229. Association for Computational Linguistics.
- Saif M. Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2015. How translation alters sentiment. *Journal of Artificial Intelligence Research*, 54:1–20.
- Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.
- Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.
- Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.
- Kashif Shah, Varvara Logacheva, Gustavo Paetzold, Frédéric Blain, Daniel Beck, Fethi Bougares, and Lucia Specia. 2015. Shef-nn: Translation quality estimation with neural networks. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 342–347, Lisbon, Portugal, September. Association for Computational Linguistics.
- Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.
- Miloš Stanojevic and Khalil Simaan. 2014. Beer: Better evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419.
- Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the wmt15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273, Lisbon, Portugal, September. Association for Computational Linguistics.
- Sara Stymne, Jörg Tiedemann, and Joakim Nivre. 2014. Estimating word alignment quality for smt reordering tasks. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 275–286, Baltimore, Maryland, USA, June. Association for Computational Linguistics.
- A. Cüneyd Tantuğ, Kemal Oflazer, and Ilknur Durgar El-Kahlout. 2008. Bleu+: a tool for fine-grained bleu computation.
- Sowmya Vajjala and Detmar Meurers. 2016. Readability-based sentence ranking for evaluating text simplification. *arXiv preprint arXiv:1603.06009*.
- Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243. Association for Computational Linguistics.
- Eray Yildiz, Caglar Tirkaz, H. Bahadır Sahin, Mustafa Tolga Eren, and Omer Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Proceedings of the AAI Conference on Artificial Intelligence*.
- Hui Yu, Qingsong Ma, Xiaofeng Wu, and Qun Liu. 2015. CASICT-DCU Participation in WMT2015 Metrics Task. *EMNLP 2015*, page 417.

Cross-Lingual Dependency Parsing with Late Decoding for Truly Low-Resource Languages

Michael Sejr Schlichtkrull

University of Amsterdam*
m.s.schlichtkrull@uva.nl

Anders Søgaard

University of Copenhagen
soegaard@di.ku.dk

Abstract

In cross-lingual dependency annotation projection, information is often lost during transfer because of early decoding. We present an end-to-end graph-based neural network dependency parser that can be trained to reproduce matrices of edge scores, which can be directly projected across word alignments. We show that our approach to cross-lingual dependency parsing is not only simpler, but also achieves an absolute improvement of 2.25% averaged across 10 languages compared to the previous state of the art.

1 Introduction

Dependency parsing is an integral part of many natural language processing systems. However, most research into dependency parsing has focused on learning from treebanks, i.e. collections of manually annotated, well-formed syntactic trees. In this paper, we develop and evaluate a graph-based parser which does not require the training data to be well-formed trees. We show that such a parser has an important application in cross-lingual learning.

Annotation projection is a method for developing parsers for low-resource languages, relying on aligned translations from resource-rich source languages into the target language, rather than linguistic resources such as treebanks or dictionaries. The Bible has been translated completely into 542 languages, and partially translated into a further 2344 languages. As such, the assumption that we have access to parallel Bible data is much less constraining than the assumption of access to linguistic resources. Furthermore, for truly low-resource languages, relying upon the Bible scales

better than relying on less biased data such as the EuroParl corpus.

In Agić et al. (2016), a projection scheme is proposed wherein labels are collected from many sources, projected into a target language, and then averaged. Crucially, the paper demonstrates how projecting and averaging edge scores from a graph-based parser *before* decoding improves performance. Even so, decoding is still a requirement between projecting labels and retraining from the projected data, since their parser (TurboParser) requires well-formed input trees. This introduces a potential source of noise and loss of information that may be important for finding the best target sentence parse.

Our approach circumvents the need for decoding prior to training, thereby surpassing a state-of-the-art dependency parser trained on decoded multi-source annotation projections as done by Agić et al. We first evaluate the model across several languages, demonstrating results comparable to the state of the art on the Universal Dependencies (McDonald et al., 2013) dataset. Then, we evaluate the same model by inducing labels from cross-lingual multi-source annotation projection, comparing the performance of a model with early decoding to a model with late decoding.

Contributions We present a novel end-to-end neural graph-based dependency parser and apply it in a cross-lingual setting where the task is to induce models for truly low-resource languages, assuming only parallel Bible text. Our parser is more flexible than similar parsers, and accepts any weighted or non-weighted graph over a token sequence as input. In our setting, the input is a dense weighted graph, and we show that our parser is superior to previous best approaches to cross-lingual parsing. The code is made available on GitHub.¹

*Work done while at the University of Copenhagen.

¹<https://github.com/MichSchli/Tensor-LSTM>

2 Model

The goal of this section is to construct a first-order graph-based dependency parser capable of learning *directly* from potentially incomplete matrices of edge scores produced by another first-order graph-based parser. Our approach is to treat the encoding stage of the parser as a tensor transformation problem, wherein tensors of edge features are mapped to matrices of edge scores. This allows our model to approximate sets of scoring matrices generated by another parser directly through non-linear regression. The core component of the model is a layered sequence of recurrent neural network transformations applied to the axes of an input tensor.

More formally, any digraph $G = (V, E)$ can be expressed as a binary $|V| \times |V|$ -matrix M , where $M_{ij} = 1$ if and only if $(j, i) \in E$ – that is, if i has an ingoing edge from j . If G is a tree rooted at v_0 , v_0 has no ingoing edges. Hence, it suffices to use a $(|V| - 1) \times |V|$ -matrix. In dependency parsing, every sentence is expressed as a matrix $S \in \mathbb{R}^{w \times f}$, where w is the number of words in the sentence and f is the width of a feature vector corresponding to each word. The goal is to learn a function $P : \mathbb{R}^{w \times f} \rightarrow \mathbb{Z}_2^{w \times (w+1)}$, such that $P(S)$ corresponds to the matrix representation of the correct parse tree for that sentence – see Figure 1 for an example.

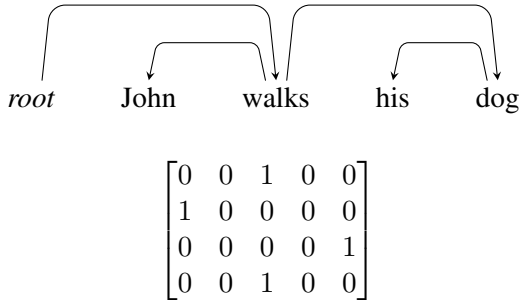


Figure 1: An example dependency tree and the corresponding parse matrix.

In the arc-factored (first-order), graph-based model, P is a composite function $P = D \circ E$ where the encoder $E : \mathbb{R}^{w \times f} \rightarrow \mathbb{R}^{w \times (w+1)}$ is a real-valued scoring function and the decoder $D : \mathbb{R}^{w \times (w+1)} \rightarrow \mathbb{Z}_2^{w \times (w+1)}$ is a minimum spanning tree algorithm (McDonald et al., 2005). Commonly, the encoder includes only *local* information – that is, E_{ij} is only dependent on S_i and

S_j , where S_i and S_j are feature vectors corresponding to dependent and head. Our contribution is the introduction of an LSTM-based *global* encoder where the entirety of S is represented in the calculation of E_{ij} .

We begin by extending S to a $(w+1) \times (f+1)$ -matrix S^* with an additional row corresponding to the root node and a single binary feature denoting whether a node is the root. We now compute a 3-tensor $F = S \boxplus S^*$ of dimension $w \times (w+1) \times (2f+1)$ consisting of concatenations of all combinations of rows in S and S^* . This tensor effectively contains a featurization of every edge (u, v) in the complete digraph over the sentence, consisting of the features of the parent word u and child word v . These edge-wise feature vectors are organized in the tensor exactly as the dependency arcs in a parse matrix such as the one shown in the example in Figure 1.

The edges represented by elements F_{ij} can as such easily be interpreted in the context of related edges represented by the row i and the column j in which that edge occurs. The classical arc-factored parsing algorithm of McDonald et al. (2005) corresponds to applying a function $O : \mathbb{R}^{2f+1} \rightarrow \mathbb{R}$ pointwise to $S \boxplus S^*$, then decoding the resulting $w \times (w+1)$ -matrix. Our model diverges by applying an LSTM-based transformation $Q : \mathbb{R}^{w \times (w+1) \times (2f+1)} \rightarrow \mathbb{R}^{w \times (w+1) \times d}$ to $S \boxplus S^*$ before applying an analogous transformation $O : \mathbb{R}_d \rightarrow \mathbb{R}$.

The Long Short-Term Memory (LSTM) unit is a function $LSTM(x, h_{t-1}, c_{t-1}) = (h_t, c_t)$ defined through the use of several intermediary steps, following Hochreiter et al. (2001). A concatenated input vector $I = x \oplus h_{prev}$ is constructed, where \oplus represents vector concatenation. Then, functions corresponding to input, forget, and output gates are defined following the form $g_{input} = \sigma(W_{input}I + b_{input})$. Finally, the internal cell state c_t and the output vector h_t at time t are defined using the Hadamard (pointwise) product \bullet :

$$\begin{aligned} c_t &= g_{forget} \bullet c_{prev} + g_{input} \bullet \tanh(W_{cell}I + b_{cell}) \\ h_t &= g_{output} \bullet \tanh(c_t) \end{aligned}$$

We define a function Matrix-LSTM inductively, that applies an LSTM to the rows of a matrix X . Formally, Matrix-LSTM is a function $\mathcal{M} : \mathbb{R}^{a \times b} \rightarrow \mathbb{R}^{a \times c}$ such that $(h_1, c_1) = LSTM(X_1, 0, 0)$, $\forall 1 < i \leq n$ $(h_i, c_i) = LSTM(X_i, h_{i-1}, c_{i-1})$, and $\mathcal{M}(X)_i = h_i$.

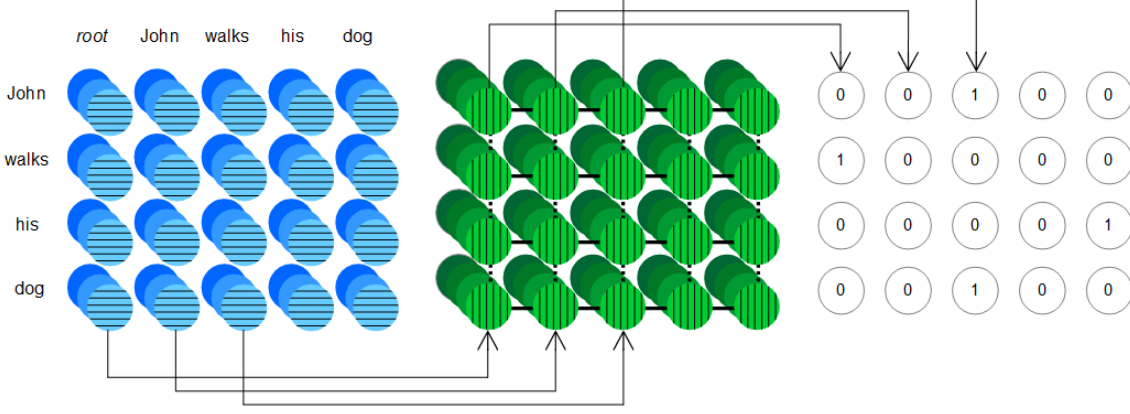


Figure 2: Four-directional Tensor-LSTM applied to the example sentence seen in Figure 1. The word-pair tensor $S \boxplus S^*$ is represented with blue units (horizontal lines), a hidden Tensor-LSTM layer H with green units (vertical lines), and the output layer with white units. The recurrent connections in the hidden layer along H and $H^{T(2,1,3)}$ are illustrated respectively with dotted and fully drawn lines.

An effective extension is the *bidirectional* LSTM, wherein the LSTM-function is applied to the sequence both in the forward and in the backward direction, and the results are concatenated. In the matrix formulation, reversing a sequence corresponds to inverting the order of the rows. This is most naturally accomplished through left-multiplication with an exchange matrix $J_m \in \mathbb{R}^{m \times m}$ such that:

$$J_m = \begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}$$

Bidirectional Matrix-LSTM is therefore defined as a function $\mathcal{M}_{2d} : \mathbb{R}^{a \times b} \rightarrow \mathbb{R}^{a \times 2c}$ such that:

$$\mathcal{M}_{2d}(S) = \mathcal{M}(S) \oplus_2 J_a \mathcal{M}(J_a S)$$

Here, \oplus_2 refers to concatenation along the second axis of the matrix.

Keeping in mind the goal of constructing a tensor transformation Q capable of propagating information in an LSTM-like manner between any two elements of the input tensor, we are interested in constructing an equivalent of the Matrix-LSTM-model operating on 3-tensors rather than matrices. This construct, when applied to the edge tensor $F = S \boxplus S^*$, can then provide a means of interpreting edges in the context of related edges.

A very simple variant of such an LSTM-function operating on 3-tensors can be constructed by applying a bidirectional Matrix-LSTM to every matrix along the first axis of the tensor. This forms

the center of our approach. Formally, bidirectional Tensor-LSTM is a function $\mathcal{T}_{2d} : \mathbb{R}^{a \times b \times c} \rightarrow \mathbb{R}^{a \times b \times 2h}$ such that:

$$\mathcal{T}_{2d}(T)_i = \mathcal{M}_{2d}(T_i)$$

This definition allows information to flow *within* the matrices of the first axis of the tensor, but not *between* them – corresponding in Figure 2 to horizontal connection along the rows, but no vertical connections along the columns. To fully cover the tensor structure, we must extend this model to include connections along columns.

This is accomplished through tensor transposition. Formally, tensor transposition is an operator $T^{T\sigma}$ where σ is a permutation on the set $\{1, \dots, \text{rank}(T)\}$. The last axis of the tensor contains the feature representations, which we are not interested in scrambling. For the Matrix-LSTM, this leaves only one option – $M^{T(1,2)}$. When the LSTM is operating on a 3-tensor, we have two options – $T^{T(2,1,3)}$ and $T^{T(1,2,3)}$. This leads to the following definition of four-directional Tensor-LSTM as a function $\mathcal{T}_{4d} : \mathbb{R}^{a \times b \times c} \rightarrow \mathbb{R}^{a \times b \times 4h}$ analogous to bidirectional Sequence-LSTM:

$$\mathcal{T}_{4d}(T) = \mathcal{T}_{2d}(T) \oplus_3 \mathcal{T}_{2d}(T^{T(2,1,3)})^{T(2,1,3)}$$

Calculating the LSTM-function on $T^{T(1,2,3)}$ and $T^{T(2,1,3)}$ can be thought of as constructing the recurrent links either "side-wards" or "down-wards" in the tensor – or, equivalently, constructing recurrent links either between the outgoing or between the in-going edges of every vertex in

the dependency graph. In Figure 2, we illustrate the two directions respectively with full or dotted edges in the hidden layer.

The output of Tensor-LSTM is itself a tensor. In our experiments, we use a multi-layered variation implemented by stacking layers of models: $\mathcal{T}_{4d,stack}(T) = \mathcal{T}_{4d}(\mathcal{T}_{4d}(\dots\mathcal{T}_{4d}(T)\dots))$. We do not share parameters between stacked layers. Training the model is done by minimizing the value $\mathcal{E}(G, O(Q(S \boxplus S^*)))$ of some loss function \mathcal{E} for each sentence S with gold tensor G . We experiment with two loss functions.

In our monolingual set-up, we exploit the fact that parse matrices by virtue of depicting trees are right stochastic matrices. Following this observation, we constrain each row of $O(Q(S \boxplus S^*))$ under a softmax-function and use as loss the row-wise cross entropy. In our cross-lingual set-up, we use mean squared error. In both cases, prediction-time decoding is done with Chu-Liu-Edmonds algorithm (Edmonds, 1968) following McDonald et al. (2005).

3 Cross-lingual parsing

Hwa et al. (2005) is a seminal paper for cross-lingual dependency parsing, but they use very detailed heuristics to ensure that the projected syntactic structures are well-formed. Agić et al. (2016) is the latest continuation of their work, presenting a new approach to cross-lingual projection, projecting edge scores rather than subtrees. Agić et al. (2016) construct target-language treebanks by aggregating scores from multiple source languages, before decoding. Averaging before decoding is especially beneficial when the parallel data is of low quality, as the decoder introduces errors, when edge scores are missing. Despite averaging, there will still be scores missing from the input weight matrices, especially when the source and target languages are very distant. Below, we show that we can circumvent error-inducing early decoding by training directly on the projected edge scores.

We assume source language datasets $\mathcal{L}_1, \dots, \mathcal{L}_n$, parsed by monolingual arc-factored parsers. In our case, this data comes from the Bible. We assume access to a set of sentence alignment functions $A_s : \mathcal{L}_s \times \mathcal{L}_t \rightarrow \mathbb{R}_{0,1}$ where $A_s(S_s, S_t)$ is the confidence that S_t is the translation of S_s . Similarly, we have access to a set of word alignment functions $W_{\mathcal{L}_s, S_s, S_t} : S_s \times S_t \rightarrow \mathbb{R}_{0,1}$ such that

$S_s \in \mathcal{L}_s$, $S_t \in \mathcal{L}_t$, and $W(w_s, w_t)$ represents the confidence that w_s aligns to w_t given that S_t is the translation of S_s

For each source language \mathcal{L}_s with a scoring function $score_{\mathcal{L}_s}$, we define a local edge-wise voting function $vote_{S_s}((u_s, v_s), (u_t, v_t))$ operating on a source language edge $(u_s, v_s) \in S_s$ and a target language edge $(u_t, v_t) \in S_t$. Intuitively, every source language edge votes for every target language edge with a score proportional to the confidence of the edges aligning and the score given in the source language. For every target language edge $(u_t, v_t) \in S_t$:

$$\begin{aligned} vote_{S_s}((u_s, v_s), (u_t, v_t)) &= W_{\mathcal{L}_s, S_s, S_t}(u_s, u_t) \\ &\quad \cdot W_{\mathcal{L}_s, S_s, S_t}(v_s, v_t) \\ &\quad \cdot score_{\mathcal{L}_s}(u_s, v_s) \end{aligned}$$

Following Agić et al. (2016), a sentence-wise voting function is then constructed as the highest contribution from a source-language edge:

$$vote_{S_s}(u_t, v_t) = \max_{u_s, v_s \in S_s} vote_{S_s}((u_s, v_s), (u_t, v_t))$$

The final contribution of each source language dataset \mathcal{L}_s to a target language edge (u_t, v_t) is then calculated as the sum for all sentences $S_s \in \mathcal{L}_s$ over $vote_{S_s}(u_t, v_t)$ multiplied by the confidence that the source language sentence aligns with the target language sentence. For an edge (u_t, v_t) in a target language sentence $S_t \in \mathcal{L}_t$:

$$vote_{\mathcal{L}_s}(u_t, v_t) = \sum_{S_s \in \mathcal{L}_s} A_s(S_s, S_t) vote_{S_s}(u_t, v_t)$$

Finally, we can compute a target language scoring function by summing over the votes for every source language:

$$score(u_t, v_t) = \frac{\sum_{i=1}^n vote_{\mathcal{L}_i}(u_t, v_t)}{Z_{S_t}}$$

Here, Z_{S_t} is a normalization constant ensuring that the target-language scores are proportional to those created by the source-language scoring functions. As such, Z_{S_t} should consist of the sum over the weights for each sentence contributing to the scoring function. We can compute this as:

$$Z_{S_t} = \sum_{i=1}^n \sum_{S_s \in \mathcal{L}_i} A_s(S_s, S_t)$$

The sentence alignment function is not a probability distribution; it may be the case that no source-language sentences contribute to a target language sentence, causing the sum of the weights *and* the sum of the votes to approach zero. In this case, we define $score(u_t, v_t) = 0$. Before projection, the source language scores are all standardized to have 0 as the mean and 1 as the standard deviation. Hence, this corresponds to assuming neither positive nor negative evidence concerning the edge.

We experiment with two methods of learning from the projected data – decoding with Chu-Liu-Edmonds algorithm and then training as proposed in Agić et al. (2016), or directly learning to reproduce the matrices of edge scores. For alignment, we use the sentence-level *hunalign* algorithm introduced in Varga et al. (2005) and the token-level model presented in Östling (2015).

4 Experiments

We conduct two sets of experiments. First, we evaluate the Tensor-LSTM-parser in the monolingual setting. We compare Tensor-LSTM to the TurboParser (Martins et al., 2010) on several languages from the Universal Dependencies dataset. In the second experiment, we evaluate Tensor-LSTM in the cross-lingual setting. We include as baselines the delexicalized parser of McDonald et al. (2011), and the approach of Agić et al. (2016) using TurboParser. To demonstrate the effectiveness of circumventing the decoding step, we conduct the cross-lingual evaluation of Tensor-LSTM using cross entropy loss with *early* decoding, and using mean squared loss with *late* decoding.

4.1 Model selection and training

Our features consist of 500-dimensional word embeddings trained on translations of the Bible. The word embeddings were trained using skipgram with negative sampling on a word-by-sentence PMI matrix induced from the Edinburgh Bible Corpus, following (Levy et al., 2017). Our embeddings are not trainable, but fixed representations throughout the learning process. Unknown tokens were represented by zero-vectors.

We combined the word embeddings with one-hot-encodings of POS-tags, projected across word alignments following the method of Agić et al. (2016). To verify the value of the POS-features, we conducted preliminary experiments on English development data. When including POS-

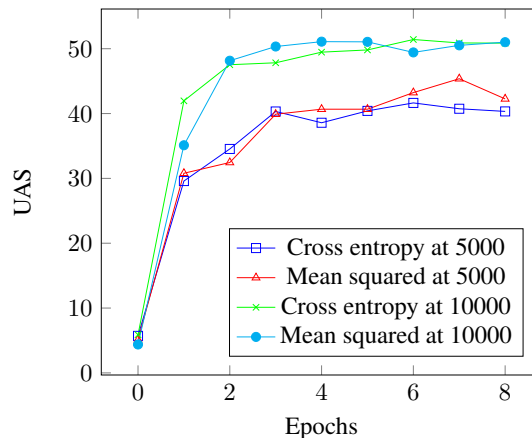


Figure 3: UAS per epoch on German development data training from 5000 or 10000 randomly sampled sentences with projected annotations.

tags, we found small, non-significant improvements for monolingual parsing, but significant improvements for cross-lingual parsing.

The weights were initialized using the normalized values suggested in Glorot and Bengio (2010). Following Jozefowicz et al. (2015), we add 1 to the initial forget gate bias. We trained the network using RMSprop (Tieleman and Hinton, 2012) with hyperparameters $\alpha = 0.1$ and $\gamma = 0.9$, using minibatches of 64 sentences. Following Neelakantan et al. (2015), we added a noise factor $n \sim \mathcal{N}(0, \frac{1}{(1+t)^{0.55}})$ to the gradient in each update. We applied dropouts after each LSTM-layer with a dropout probability $p = 0.5$, and between the input layer and the first LSTM-layer with a dropout probability of $p = 0.2$ (Bluche et al., 2015). As proposed in Pascanu et al. (2012), we employed a gradient clipping factor of 15. In the monolingual setting, we used early stopping on the development set.

We experimented with 10, 50, 100, and 200 hidden units per layer, and with up to 6 layers. Using greedy search on monolingual parsing and evaluating on the English development data, we determined the optimal network shape to contain 100 units per direction per hidden layer, and a total of 4 layers.

For the cross-lingual setting, we used two additional hyper-parameters. We used the development data from one of our target languages (German) to determine the optimal number of epochs before stopping. Furthermore, we trained only on a subset of the projected sentences, choosing the size of the subset using the development data.

We experimented with either 5000 or 10000 randomly sampled sentences. There are two motivating factors behind this subsampling. First, while the Bible in general consists of about 30000 sentences, for many low-resource languages we do not have access to annotation projections for the full Bible, because parts were never translated, and because of varying projection quality. Second, subsampling speeds up the training, which was necessary to make our experiments practical: At 10000 sentences and on a single GPU, each epoch takes approximately 2.5 hours. As such, training for a single language could be completed in less than a day. We plot the results in Figure 3. We see that the best performance is achieved at 10000 sentences, and with respectively 6 and 5 epochs for cross entropy and mean squared loss.

4.2 Results

In the monolingual setting, we compare our parser to TurboParser (Martins et al., 2010) – a fast, capable graph-based parser used as a component in many larger systems. TurboParser is also the system of choice for the cross-lingual pipeline of Agić et al. (2016). It is therefore interesting to make a direct comparison between the two. The results can be seen in Table 1.

Language	TurboParser	Tensor-LSTM
English*	83.84	85.81
German	81.45	82.64
Danish	81.82	82.24
Finnish	77.74	78.83
Spanish	83.19	86.69
French	81.17	84.63
Czech	81.32	85.04
Average	81.50	83.70

Table 1: Unlabeled Attachment Score on the UD test data for TurboParser and Tensor-LSTM with cross entropy loss. English development data was used for model selection (marked *).

Note that in order for a parser to be directly applicable to the annotation projection setup explored in the secondary experiment, it must be a *first-order graph-based* parser. In the monolingual setting, the best results reported so far (84.74, on average) for the above selection of treebanks were by the Parsito system (Straka et al., 2015), a transition-based parser using a dynamic oracle.

For the cross-lingual annotation projection experiments, we use the delexicalized system suggested by McDonald et al. (2011) as a baseline. We also compare against the annotation projection scheme using TurboParser suggested in Agić et al. (2016), representing the previous state of the art for truly low-resource cross-lingual dependency parsing. Note that while our results for the TurboParser-based system use the same training data, test data, and model as in Agić et al., our results differ due to the use of the Bible corpus rather than a Watchtower publications corpus as parallel data. The authors made results available using the Edinburgh Bible Corpus for unlabeled data. The two tested conditions of Tensor-LSTM are the mean squared loss model *without* intermediary decoding, and the cross entropy model *with* intermediary decoding. The results of the cross-lingual experiment can be seen in Table 2.

5 Discussion

As is evident from Table 2, the variation in performance across different languages is large for all systems. This is to be expected, as the quality of the projected label sets vary widely due to linguistic differences. On average, Tensor-LSTM with mean squared loss outperforms all other systems. In Section 1, we hypothesized that incomplete projected scorings would have a larger impact upon systems reliant on an intermediary decoding step. To investigate this claim, we plot in Figure 4 the performance difference with mean squared loss and cross entropy loss for each language versus the percentage of missing edge scores.

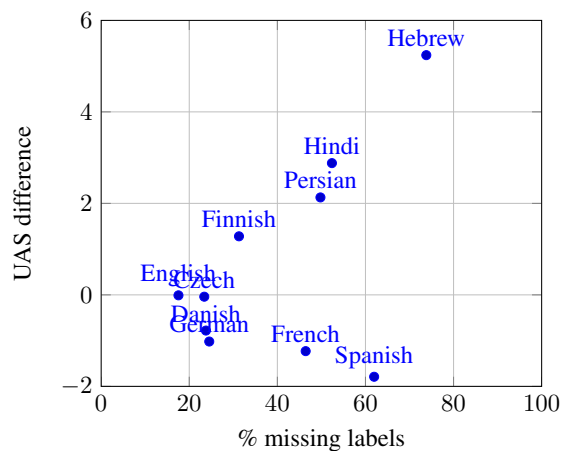


Figure 4: Percentage of missing edge scores versus performance difference for Tensor-LSTM with mean squared loss and cross entropy loss.

Language	Delexicalized	TurboParser	Tensor-LSTM (Decoding)	Tensor-LSTM (No decoding)
Czech (cs)	40.99	43.81	42.58	41.54
Danish (da)	49.65	54.87	54.93	54.15
English* (en)	48.08	52.52	52.91	52.90
Finnish (fi)	41.18	46.08	43.98	45.26
French (fr)	48.97	45.83	55.06	53.83
German* (de)	49.36	51.79	54.87	53.85
Spanish (es)	47.60	58.90	59.60	57.81
Persian (fa)	28.93	14.88	46.47	48.60
Hebrew (he)	19.06	52.89	26.17	31.41
Hindi (hi)	21.03	43.31	43.21	46.09
Average	39.49	46.29	47.98	48.54

Table 2: Unlabeled attachment scores for the various systems. Tensor-LSTM is evaluated using cross entropy and mean squared loss. We include the results of two baselines – the delexicalized system of McDonald et al. (2011) and the Turbo-based projection scheme of Agić et al. (2016). English and German development data was used for hyperparameter tuning (marked *).

For languages outside the Germanic and Latin families, our claim holds – the performance of the cross entropy loss system decreases faster with the percentage of missing labels than the performance of the mean squared loss system. To an extent, this confirms our hypothesis, as we for the average language observe an improvement by circumventing the decoding step. French and Spanish, however, do not follow the same trend, with cross entropy loss outperforming mean squared loss despite the high number of missing labels.

In Table 2, performance on French and Spanish for both systems can be seen to be very high. It may be the case that Indo-European target languages are not as affected by missing labels as most of the *source* languages are themselves Indo-European. Another explanation could be that some feature of the cross entropy loss function makes it especially well suited for Latin languages – as seen in Table 1, French and Spanish are also two of the languages for which Tensor-LSTM yields the highest performance improvement.

To compare the effect of missing edge scores upon performance without influence from linguistic factors such as language similarity, we repeat the cross-lingual experiment on one language with respectively 10%, 20%, 30%, and 40% of the projected and averaged edge scores artificially set to 0, simulating missing data. We choose the English data for this experiment, as the English projected data has the lowest percentage of missing labels

across any of the languages. In Figure 5, we plot the performance for each of the two systems versus the percentage of deleted values.

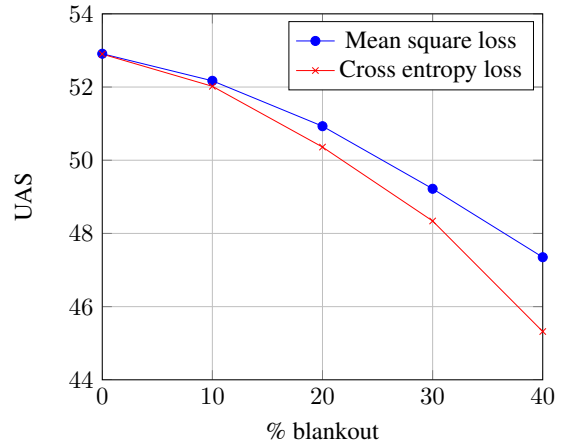


Figure 5: Performance for Tensor-LSTM on English test data with 0-40% of the edge scores artificially maintained at 0.

As can be clearly seen, performance drops faster with the percentage of deleted labels for the cross entropy model. This confirms our intuition that the initially lower performance using mean squared loss compared to cross entropy loss is mitigated by a greater robustness towards missing labels, gained by circumventing the decoding step in the training process. In Table 2, this is reflected as dramatic performance increases using mean squared error for Finnish, Persian, Hindi, and Hebrew – the four languages furthest

removed from the predominantly Indo-European source languages and therefore the four languages with the poorest projected label quality.

Several possible avenues for future work on this project are available. In this paper, we used an extremely simple feature function. More complex feature functions is one potential source of improvement. Another interesting direction for future work would be to include POS-tagging directly as a component of Tensor-LSTM prior to the construction of $S \boxplus S^*$ in a multi-task learning framework. Similarly, incorporating semantic tasks on top of dependency parsing could lead to interesting results. Finally, extensions of the Tensor-LSTM function to deeper models, wider models, or more connected models as seen in e.g. Kalchbrenner et al. (2015) may yield further performance gains.

6 Related Work

Experiments with neural networks for dependency parsing have focused mostly on learning higher-order scoring functions and creating efficient feature representations, with the notable exception of Fonseca et al. (2015). In their paper, a convolutional neural network is used to evaluate local edge scores based on global information. In Zhang and Zhao (2015) and Pei et al. (2015), neural networks are used to simultaneously evaluate first-order and higher-order scores for graph-based parsing, demonstrating good results. Bidirectional LSTM-models have been successfully applied to feature generation (Kiperwasser and Goldberg, 2016). Such LSTM-based features could in future work be employed and trained in conjunction with Tensor-LSTM, incorporating global information both in parsing and in featurization.

An extension of LSTM to tensor-structured data has been explored in Graves et al. (2007), and further improved upon in Kalchbrenner et al. (2015) in the form of GridLSTM. Our approach is similar, but simpler and computationally more efficient as no within-layer connections between the first and the second axes of the tensor are required.

Annotation projection for dependency parsing has been explored in a number of papers, starting with Hwa et al. (2005). In Tiedemann (2014) and Tiedemann (2015) the process is extended and evaluated across many languages. Li et al. (2014) follows the method of Hwa et al. (2005) and adds a probabilistic target-language classifier to deter-

mine and filter out high-uncertainty trees. In Ma and Xia (2014), performance on projected data is used as an additional objective for unsupervised learning through a combined loss function.

A common thread in these papers is the use of high-quality parallel data such as the EuroParl corpus. For truly low-resource target languages, this setting is unrealistic as parallel resources may be restricted to biased data such as the Bible. In Agić et al. (2016) this problem is addressed, and a parser is constructed which utilizes averaging over edge posteriors for many source languages to compensate for low-quality projected data. Our work builds upon their contribution by constructing a more flexible parser which can bypass a source of bias in their projected labels, and we therefore compared our results directly to theirs.

Annotation projection procedures for cross-lingual dependency parsing has been the focus of several other recent papers (Guo et al., 2015; Zhang and Barzilay, 2015; Duong et al., 2015; Rasooli and Collins, 2015). In Guo et al. (2015), distributed, language-independent feature representations are used to train shared parsers. Zhang and Barzilay (2015) introduce a tensor-based feature representation capable of incorporating prior knowledge about feature interactions learned from source languages. In Duong et al. (2015), a neural network parser is built wherein higher-level layers are shared between languages.

Finally, Rasooli and Collins (2015) leverage dense information in high-quality sentence translations to improve performance. Their work can be seen as opposite to ours – whereas Rasooli and Collins leverage high-quality translations to improve performance when such are available, we focus on improving performance in the *absence* of high-quality translations.

7 Conclusion

We have introduced a novel algorithm for graph-based dependency parsing based on an extension of sequence-LSTM to the more general Tensor-LSTM. We have shown how the parser with a cross entropy loss function performs comparably to state of the art for monolingual parsing. Furthermore, we have demonstrated that the flexibility of our parser enables learning from non well-formed data and from the output of other parsers. Using this property, we have applied our parser to a cross-lingual annotation projection problem

for truly low-resource languages, demonstrating an average target-language unlabeled attachment score of 48.54, which to the best of our knowledge are the best results yet for the task.

Acknowledgments

The second author was supported by ERC Starting Grant No. 313695.

References

- Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4.
- Theodore Bluche, Christopher Kermorvant, and Jerome Louradour. 2015. Where to apply dropout in recurrent neural networks for handwriting recognition? In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 681–685. IEEE.
- Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 845–850. Association for Computational Linguistics.
- Jack Edmonds. 1968. Optimum branchings. In *Mathematics and the Decision Sciences, Part 1*, pages 335–345. American Mathematical Society.
- Erick R. Fonseca, Avenida Trabalhador São-carlense, and Sandra M. Aluísio. 2015. A deep architecture for non-projective dependency parsing. In *Proceedings of the 2015 NAACL-HLT Workshop on Vector Space Modeling for NLP*, pages 56–61. Association for Computational Linguistics.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 2010 International conference on Artificial Intelligence and Statistics*, pages 249–256. Society for Artificial Intelligence and Statistics.
- Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. *arXiv preprint arXiv:0705.2011*.
- Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.
- Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamic Recurrent Neural Networks*. IEEE press.
- Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350. International Machine Learning Society.
- Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.
- Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word representations from sentence alignments. In *EACL*.
- Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Soft cross-lingual syntax projection for dependency parsing. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 783–793. Association for Computational Linguistics.
- Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1337–1348. Association for Computational Linguistics.
- André F.T. Martins, Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.
- Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on*

- Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics.
- Ryan T. McDonald, Joakim Nivre, Yvonne Quirnbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B. Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.
- Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*.
- Robert Östling. 2015. *Bayesian models for multilingual word alignment*. Ph.D. thesis, Department of Linguistics, Stockholm University.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.
- Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 313–322. Association for Computational Linguistics.
- Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338. Association for Computational Linguistics.
- Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. 2015. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of the 14th International Workshop on Treebanks and Linguistic Theories*, pages 208–220. Association for Computational Linguistics.
- Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864. Association for Computational Linguistics.
- Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. *Proceedings of the Third International Conference on Dependency Linguistics*, pages 340–349.
- Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:2.
- Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2005. Parallel corpora for medium density languages. In *Proceedings of the 2005 Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics.
- Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867. Association for Computational Linguistics.
- Zhisong Zhang and Hai Zhao. 2015. High-order graph-based neural dependency parsing. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 114–123. Association for Computational Linguistics.

Parsing Universal Dependencies without training

Héctor Martínez Alonso[♣] Željko Agić[♡] Barbara Plank[♣] Anders Søgaard[◇]

[♣]Univ. Paris Diderot, Sorbonne Paris Cité – Alpage, INRIA, France

[♡]IT University of Copenhagen, Denmark

[♣]Center for Language and Cognition, University of Groningen, The Netherlands

[◇]University of Copenhagen, Denmark

hector.martinez-alonso@inria.fr, zeag@itu.dk, b.plank@rug.nl, soegaard@di.ku.dk

Abstract

We propose UDP, the first training-free parser for Universal Dependencies (UD). Our algorithm is based on PageRank and a small set of head attachment rules. It features two-step decoding to guarantee that function words are attached as leaf nodes. The parser requires no training, and it is competitive with a delexicalized transfer system. UDP offers a linguistically sound unsupervised alternative to cross-lingual parsing for UD, which can be used as a baseline for such systems. The parser has very few parameters and is distinctly robust to domain change across languages.

1 Introduction

Grammar induction and unsupervised dependency parsing are active fields of research in natural language processing (Klein and Manning, 2004; Gelling et al., 2012). However, many data-driven approaches struggle with learning relations that match the conventions of the test data, e.g., Klein and Manning reported the tendency of their DMV parser to make determiners the heads of German nouns, which would not be an error if the test data used a DP analysis (Abney, 1987). Even supervised transfer approaches (McDonald et al., 2011) suffer from target adaptation problems when facing word order differences.

The Universal Dependencies (UD) project (Nivre et al., 2015; Nivre et al., 2016) offers a dependency formalism that aims at providing a consistent representation across languages, while enforcing a few hard constraints. The arrival of such treebanks, expanded and improved on a regular basis, provides a new milestone for cross-lingual dependency parsing research (McDonald et al., 2013).

Furthermore, given that UD rests on a series of simple principles like the primacy of lexical heads, cf. Johannsen et al. (2015) for more details, we expect that such a formalism lends itself more naturally to a simple and linguistically sound rule-based approach to cross-lingual parsing. In this paper we present such an approach.

Our system is a dependency parser that requires no training, and relies solely on explicit part-of-speech (POS) constraints that UD imposes. In particular, UD prescribes that trees are single-rooted, and that function words like adpositions, auxiliaries, and determiners are always dependents of content words, while other formalisms might treat them as heads (De Marneffe et al., 2014). We ascribe our work to the viewpoints of Bender (2009) about the incorporation of linguistic knowledge in language-independent systems.

Contributions We introduce, to the best of our knowledge, the first unsupervised rule-based dependency parser for Universal Dependencies.

Our method goes substantially beyond the existing work on *rule-aided* unsupervised dependency parsing, specifically by:

- i) adapting the dependency head rules to UD-compliant POS relations,
- ii) incorporating the UD restriction of function words being leaves,
- iii) applying personalized PageRank to improve main predicate identification, and by
- iv) making the parsing entirely free of language-specific parameters by estimating adposition attachment direction at runtime.

We evaluate our system on 32 languages¹ in three setups, depending on the reliability of available POS tags, and compare to a multi-source delexi-

¹Out of 33 languages in UD v1.2. We exclude Japanese because the treebank is distributed without word forms and hence we can not provide results on predicted POS.

calized transfer system. In addition, we evaluate the systems’ sensitivity to domain change for a subset of UD languages for which domain information was retrievable. The results expose a solid and competitive system for all UD languages. Our unsupervised parser compares favorably to delexicalized parsing, while being more robust to domain change.

2 Related work

Cross-lingual learning Recent years have seen exciting developments in cross-lingual linguistic structure prediction based on transfer or projection of POS and dependencies (Das and Petrov, 2011; McDonald et al., 2011). These works mainly use supervised learning and domain adaptation techniques for the target language.

The first group of approaches deals with annotation projection (Yarowsky et al., 2001), whereby parallel corpora are used to transfer annotations between resource-rich source languages and low-resource target languages. Projection relies on the availability and quality of parallel corpora, source-side taggers and parsers, but also tokenizers, sentence aligners, and word aligners for sources and targets. Hwa et al. (2005) were the first to project syntactic dependencies, and Tiedemann et al. (2014; 2016) improved on their projection algorithm. Current state of the art in cross-lingual dependency parsing involves leveraging parallel corpora for annotation projection (Ma and Xia, 2014; Rasooli and Collins, 2015).

The second group of approaches deals with transferring source parsing models to target languages. Zeman and Resnik (2008) were the first to introduce the idea of delexicalization: removing lexical features by training and cross-lingually applying parsers solely on POS sequences. Sjøgaard (2011) and McDonald et al. (2011) independently extended the approach by using multiple sources, requiring uniform POS and dependency representations (McDonald et al., 2013).

Both model transfer and annotation projection rely on a large number of presumptions to derive their competitive parsing models. By and large, these presumptions are unrealistic and exclusive to a group of very closely related, resource-rich Indo-European languages. Agić et al. (2015; 2016) exposed some of these biases in their proposal for realistic cross-lingual tagging and parsing, as they emphasized the lack of perfect sentence- and

word-splitting for truly low-resource languages. Further, Johannsen et al. (2016) introduced joint projection of POS and dependencies from multiple sources while sharing the outlook on bias removal in real-world multilingual processing.

Rule-based parsing Cross-lingual methods, realistic or not, depend entirely on the availability of data: for the sources, for the targets, or most often for both sets of languages. Moreover, they typically do not exploit constraints placed on linguistic structures through a formalism, and they do so *by design*.

With the emergence of UD as the practical standard for multilingual POS and syntactic dependency annotation, we argue for an approach that takes a fresh angle on both aspects. Specifically, we propose a parser that i) requires *no* training data, and in contrast ii) critically *relies* on exploiting the UD constraints.

These two characteristics make our parser unsupervised. Data-driven unsupervised dependency parsing is now a well-established discipline (Klein and Manning, 2004; Spitkovsky et al., 2010a; Spitkovsky et al., 2010b). Still, the performance of these parsers falls far behind the approaches involving any sort of supervision.

Our work builds on the line of research on rule-aided unsupervised dependency parsing by Gillenwater et al. (2010) and Naseem et al. (2010), and also relates to Sjøgaard’s (2012a; 2012b) work. Our parser, however, features two key differences:

- i) the usage of PageRank personalization (Lofgren, 2015), and of
- ii) two-step decoding to treat content and function words differently according to the UD formalism.

Through these differences, even without any training data, we parse nearly as well as a delexicalized transfer parser, and with increased stability to domain change.

3 Method

Our approach does not use any training or unlabeled data. We have used the English treebank during development to assess the contribution of individual head rules, and to tune PageRank parameters (Sec. 3.1) and function-word directionality (Sec. 3.2). Adposition direction is calculated on the fly at runtime. We refer henceforth to our UD parser as UDP.

3.1 PageRank setup

Our system uses the PageRank (PR) algorithm (Page et al., 1999) to estimate the relevance of the content words of a sentence. PR uses a random walk to estimate which nodes in the graph are more likely to be visited often, and thus, it gives higher rank to nodes with more incoming edges, as well as to nodes connected to those. Using PR to score word relevance requires an effective graph-building strategy. We have experimented with the strategies by Søgaard (2012b), such as words being connected to adjacent words, but our system fares best strictly using the dependency rules in Table 1 to build the graph. UD trees are often very flat, and a highly connected graph yields a PR distribution that is closer to uniform, thereby removing some of the difference of word relevance.

We build a multigraph of all words in the sentence covered by the head-dependent rules in Table 1, giving each word an incoming edge for each eligible dependent, i.e., ADV depends on ADJ and VERB. This strategy does not always yield connected graphs, and we use a teleport probability of 0.05 to ensure PR convergence.

Teleport probability is the probability that, in any iteration of the PR calculation, the next active node is randomly chosen, instead of being one of the adjacent nodes of the current active node. See Brin and Page (1998) for more details on teleport probability, where the authors refer to one minus teleport probability as *damping factor*.

We chose this value incrementally in intervals of 0.01 during development until we found the smallest value that guaranteed PR convergence. A high teleport probability is undesirable, because the resulting stationary distribution can be almost uniform. We did not have to re-adjust this value when running on the actual test data.

The main idea behind our personalized PR approach is the observation that ranking is only relevant for content words.² PR can incorporate a priori knowledge of the relevance of nodes by means of *personalization*, namely giving more weight to certain nodes.

Intuitively, the higher the rank of a word, the closer it should be to the root node, i.e., the main predicate of the sentence is the node that should have the highest PR, making it the dependent of the root node (Fig. 1, lines 4-5). We use PR personalization to give 5 times more weight (over an

```

1:  $H = \emptyset; D = \emptyset$ 
2:  $C = \langle c_1, \dots, c_m \rangle; F = \langle f_1, \dots, f_m \rangle$ 
3: for  $c \in C$  do
4:   if  $|H| = 0$  then
5:      $h = root$ 
6:   else
7:      $h = \operatorname{argmin}_{j \in H} \{ \gamma(j, c) \mid \delta(j, c) \wedge \kappa(j, c) \}$ 
8:   end if
9:    $H = H \cup \{c\}$ 
10:   $D = D \cup \{(h, c)\}$ 
11: end for
12: for  $f \in F$  do
13:   $h = \operatorname{argmin}_{j \in H} \{ \gamma(j, f) \mid \delta(j, f) \wedge \kappa(j, f) \}$ 
14:   $D = D \cup \{(h, f)\}$ 
15: end for
16: return  $D$ 

```

Figure 1: Two-step decoding algorithm for UDP.

otherwise uniform distribution) to the node that is estimated to be main predicate, i.e., the first verb or the first content word if there are no verbs.

3.2 Head direction

Head direction is an important trait in dependency syntax (Tesnière, 1959). Indeed, the UD feature inventory contains a trait to distinguish the general adposition tag ADP in pre- and post-positions.

Instead of relying on this feature from the treebanks, which is not always provided, we estimate the frequency of ADP-NOMINAL vs. NOMINAL-ADP bigrams.³ We calculate this estimation directly on input data at runtime to keep the system training-free. Moreover, it requires very few examples to converge (10-15 sentences). If a language has more ADP-NOMINAL bigrams, we consider all its ADP to be prepositions (and thus dependent of elements at their right). Otherwise, we consider them postpositions.

For other function words, we have determined on the English dev data whether to make them strictly right- or left-attaching, or to allow either direction. There, AUX, DET, and SCONJ are right-attaching, while CONJ and PUNCT are left-attaching. There are no direction constraints for the rest. Punctuation is a common source of parsing errors that has very little interest in this setup. While we do evaluate on all tokens including punctuation, we also apply a heuristic for the last token in a sentence; if it is a punctuation, we make it a dependent of the main predicate.

²ADJ, NOUN, PROPN, and VERB mark content words.

³NOMINAL = {NOUN, PROPN, PRON}

ADJ	→	ADV
NOUN	→	ADJ, NOUN, PROPN
NOUN	→	ADP, DET, NUM
PROPN	→	ADJ, NOUN, PROPN
PROPN	→	ADP, DET, NUM
VERB	→	ADV, AUX, NOUN
VERB	→	PROPN, PRON, SCONJ

Table 1: UD dependency rules.

3.3 Decoding

Fig. 1 shows the tree-decoding algorithm. It has two blocks, namely a first block (3-11) where we assign the head of content words according to their PageRank and the constraints of the dependency rules, and a second block (12-15) where we assign the head of function words according to their proximity, direction of attachment, and dependency rules. The algorithm requires:

1. The PR-sorted list of content words C .
2. The set of function words F , sorting is irrelevant because function-head assignments are inter-independent.
3. A set H for the current possible heads, and a set D for the dependencies assigned at each iteration, which we represent as head-dependent tuples (h, d) .
4. A symbol $root$ for the root node.
5. A function $\gamma(n, m)$ that gives the linear distance between two nodes.
6. A function $\kappa(h, d)$ that returns whether the dependency (h, d) has a valid attachment direction given the POS of the d (cf. Sec. 3.2).
7. A function $\delta(h, d)$ that determines whether (h, d) is licensed by the rules in Table 1.

The head assignments in lines 7 and 13 read as follow: the head h of a word (either c or f) is the closest element of the current list of heads (H) that has the right direction (κ) and respects the POS-dependency rules (δ). These assignments have a back-off option to ensure the final D is a tree. If the conditions determined by κ and δ are too strict, i.e., if the set of possible heads is empty, we drop the δ head-rule constraint and recalculate the closest possible head that respects the directionality imposed by κ . If the set is empty again, we drop both constraints and assign the closest head.

Lines 4 and 5 enforce the single-root constraint. To enforce the leaf status of function nodes, the algorithm first attaches all content words (C), and

then all function words (F) in the second block where H is not updated, thereby ensuring leafness for all $f \in F$. The order of head attachment is not monotonic wrt. PR between the first and second block, and can yield non-projectivities. Nevertheless, it still is a one-pass algorithm. Decoding runs in less than $O(n^2)$, namely $O(n \times |C|)$. However, running PR incurs the main computation cost.

4 Parser run example

This section exemplifies a full run of UDP for the example sentence from the English test data: “They also had a special connection to some extremists”.

4.1 PageRank

Given an input sentence and its POS tags, we obtain rank of each word by building a graph using head rules and running PR on it. Table 2 provides the sentence, the POS of each word, the number of incoming edges for each word after building the graph with the head rules from Sec. 3.1, and the personalization vector for PR on this sentence. Note that all nodes have the same personalization weight, except the estimated main predicate, the verb “had”.

Word:	They	also	had	a	special	connection	to	some	extremists
POS:	PRON	ADV	VERB	DET	ADJ	NOUN	ADP	DET	NOUN
Personalization:	1	1	5	1	1	1	1	1	1
Incoming edges:	0	0	4	0	1	5	0	0	5

Table 2: Words, POS, personalization, and incoming edges for the example sentence.

Table 4 shows the directed multigraph used for PR in detail. We can see, e.g., that the four incoming edges for the verb “had” from the two nouns, plus from the adverb “also” and the pronoun “They”.

After running PR, we obtain the following ranking for content words:

$$C = \langle \text{had}, \text{connection}, \text{extremists}, \text{special} \rangle$$

Even though the verb has four incoming edges and the nouns have five each, the personalization makes the verb the highest-ranked word.

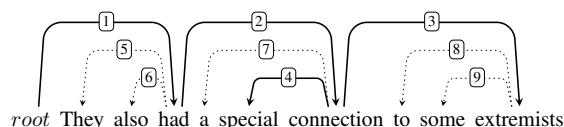


Figure 2: Example dependency tree predicted by the algorithm.

4.2 Decoding

Once C is calculated, we can follow the algorithm in Fig. 1 to obtain a dependency parse. Table 3 shows a trace of the algorithm, with $C = \langle \text{had,connection,extremists,special} \rangle$ and $F = \{\text{They,also,a,to,some}\}$.

it	$word$	h	H
1	had	root	\emptyset
2	connection	had	$\{\text{had}\}$
3	extremists	had	$\{\text{had,connection}\}$
4	special	connection	$\{\text{had,connection,extremists}\}$
5	They	had	$\{\text{had,connection,extremists,special}\}$
6	also	had	...
7	a	connection	...
8	to	extremists	...
9	some	extremists	...

Table 3: Algorithm trace for example sentence. it : iteration number, $word$: current word, H : set of possible heads.

The first four iterations calculate the head of content words following their PR, and the following iterations attach the function words in F . Finally, Fig. 2 shows the resulting dependency tree. Full lines are assigned in the first block (content dependents), dotted lines are assigned in the second block (function dependents). The edge labels indicate in which iteration the algorithm has assigned each dependency. Note that the algorithm is deterministic for a certain input POS sequence. Any 10-token sentence with the POS labels shown in Table 2 would yield the same dependency tree.⁴

5 Experiments

This section describes the data, metrics and comparison systems used to assess the performance of UDP. We evaluate on the test sections of the UD1.2 treebanks (Nivre et al., 2015) that contain word forms. If there is more than one treebank per language, we use the treebank that has the

⁴The resulting trees always pass the validation script in github.com/UniversalDependencies/tools.

\rightarrow	They	also	had	a	special	connection	to	some	extremists
They	-		•				•		
also		-	•	•					
had			-						
a				-		•			•
special					-	•			•
connection			•			-			•
to							-		•
some								-	•
extremists			•			•			-

Table 4: Matrix representation of the directed graph for the words in the sentence.

canonical language name (e.g., *Finnish* instead of *Finnish-FTB*). We use standard unlabeled attachment score (UAS) and evaluate on all sentences of the canonical UD test sets.

5.1 Baseline

We compare our UDP system with the performance of a rule-based baseline that uses the head rules in Table 5. The baseline identifies the first verb (or first content word if there are no verbs) as the main predicate, and assigns heads to all words according to the rules in Table 1. We have selected the set of head rules to maximize precision on the development set, and they do not provide full coverage. The system makes any word not covered by the rules (e.g., a word with a POS such as X or SYM) either dependent of their left or right neighbor, according to the estimated runtime parameter.

We report the best head direction and its score for each language in Table 5. This baseline finds the head of each token based on its closest possible head, or on its immediate left or right neighbor if there is no head rule for the POS at hand, which means that this system does not necessarily yield well-formed trees. Each token receives a head, and while the structures are single-rooted, they are not necessarily connected. Note that we do not include results for the DMV model by Klein and Manning (2004), as it has been outperformed by a system similar to ours (Søgaard, 2012b). The usual adjacency baseline for unsupervised dependency parsing, where all words depend on their left or right neighbor, fares much worse than our baseline (20% UAS below on average) even with an oracle pick for the best per-language direction, and we do not report those scores.

5.2 Evaluation setup

Our system relies solely on POS tags. To estimate the quality degradation of our system under non-gold POS scenarios, we evaluate UDP on two alternative scenarios. The first is predicted POS (UDP_P), where we tag the respective test set with TnT (Brants, 2000) trained on each language’s training set. The second is a naive type-constrained two-POS tag scenario (UDP_N), and approximates a lower bound. We give each word either CONTENT or FUNCTION tag, depending on the word’s frequency. The 100 most frequent words of the input test section receive the FUNCTION tag.

Finally, we compare our parser UDP to a supervised cross-lingual system (MSD). It is a multi-source delexicalized transfer parser, referred to as *multi-dir* in the original paper by McDonald et al. (2011). For this baseline we train TurboParser (Martins et al., 2013) on a delexicalized training set of 20k sentences, sampled uniformly from the UD training data excluding the target language. MSD is a competitive and realistic baseline in cross-lingual transfer parsing work. This gives us an indication how our system compares to standard cross-lingual parsers.

5.3 Results

Table 5 shows that UDP is a competitive system; because UDP_G is remarkably close to the supervised MSD_G system, with an average difference of 6.4%. Notably, UDP even outperforms MSD on one language (Hindi).

More interestingly, on the evaluation scenario with predicted POS we observe that our system drops only marginally (2.2%) compared to MSD (2.7%). In the least robust rule-based setup, the error propagation rate from POS to dependency would be doubled, as either a wrongly tagged head or dependent would break the dependency rules. However, with an average POS accuracy by TnT of 94.1%, the error propagation is 0.37, i.e., each POS error causes 0.37 additional dependency errors. In contrast, for MSD this error propagation is 0.46, thus higher.⁵

For the extreme POS scenario, content vs. function POS (CF), the drop in performance for UDP is very large, but this might be too crude an evaluation setup. Nevertheless, UDP, the simple unsupervised system with PageRank, outperforms the adjacency baselines (BL) by $\sim 4\%$ on average on the two type-based naive POS tag scenario. This difference indicates that even with very deficient POS tags, UDP can provide better structures.

6 Discussion

In this section we provide a further error analysis of the UDP parser. We examine the contribution to the overall results of using PageRank to score content words, the behavior of the system across different parts of speech, and we assess the robustness of UDP on text from different domains.

⁵Err. prop. = $(E(Parse_P) - E(Parse_G)) / E(POS_P)$, where $E(x) = 1 - Accuracy(x)$.

Language	BL _G	UDP _G	MSD _G	MSD _P	UDP _P	UDP _N
Ancient Greek	42.2 L	43.4	48.6	46.5	41.6	27.0
Arabic	34.8 R	47.8	52.8	52.6	47.6	41.0
Basque	47.8 R	45.0	51.2	49.3	43.1	22.8
Bulgarian	54.9 R	70.5	78.7	76.6	68.1	27.1
Church Slavonic	53.8 L	59.2	61.8	59.8	59.2	35.2
Croatian	41.6 L	56.7	69.1	65.6	54.5	25.2
Czech	46.5 R	61.0	69.5	67.6	59.3	25.3
Danish	47.3 R	57.9	70.2	65.6	53.8	26.9
Dutch	36.1 L	49.5	57.0	59.2	50.0	24.1
English	46.2 R	53.0	62.1	59.9	51.4	27.9
Estonian	<u>73.2</u> R	70.0	73.4	66.1	65.0	25.3
Finnish	43.8 R	45.1	52.9	50.4	43.1	21.6
French	47.1 R	64.5	72.7	70.6	62.1	36.3
German	48.2 R	60.6	66.9	62.5	57.0	24.2
Gothic	50.2 L	57.5	61.7	59.2	55.8	34.1
Greek	45.7 R	58.5	68.0	66.4	57.0	29.3
Hebrew	41.8 R	55.4	62.0	58.6	52.8	35.7
Hindi	43.9 R	46.3	34.6	34.5	45.7	27.0
Hungarian	53.1 R	56.7	58.4	56.8	54.8	22.7
Indonesian	44.6 L	60.6	63.6	61.0	58.4	35.3
Irish	47.5 R	56.6	62.5	61.3	53.9	35.8
Italian	50.6 R	69.4	77.1	75.2	67.9	37.6
Latin	49.4 L	56.2	59.8	54.9	52.4	37.1
Norwegian	49.1 R	61.7	70.8	67.3	58.6	29.8
Persian	37.8 L	55.7	57.8	55.6	53.6	33.9
Polish	60.8 R	68.4	75.6	71.7	65.7	34.6
Portuguese	45.8 R	65.7	72.8	71.4	64.9	33.5
Romanian	52.7 R	63.7	69.2	64.0	58.9	32.1
Slovene	50.6 R	63.6	74.7	71.0	56.0	24.3
Spanish	48.2 R	63.9	72.9	70.7	62.1	35.0
Swedish	52.4 R	62.8	72.2	67.2	58.5	25.3
Tamil	<u>41.4</u> R	34.2	44.2	39.5	32.1	20.3
Average	47.8	57.5	63.9	61.2	55.3	29.9

Table 5: UAS for baseline with gold POS (BL_G) with direction (L/R) for backoff attachments, UDP with gold POS (UDP_G) and predicted POS (UDP_P), PR with naive content-function POS (UDP_N), and multi-source delexicalized with gold and predicted POS (MSD_G and MSD_P, respectively). BL values higher than UDP_G are underlined, and UDP_G values higher than MSD_G are in boldface.

6.1 PageRank contribution

UDP depends on PageRank to score content words, and on two-step decoding to ensure the leaf status of function words. In this section we isolate the contribution of both parts. We do so by comparing the performance of BL, UDP, and UDP_{NoPR}, a version of UDP where we disable PR and rank content words according to their reading order, i.e., the first word in the ranking is the first word to be read, regardless of the specific language’s script direction. The baseline BL described in 5.1 already ensures function words are leaf nodes, because they have no listed dependent POS in the head rules. The task of the decoding steps is mainly to ensure the resulting structures are well-formed dependency trees.

If we measure the difference between UDP_{NoPR} and BL, we see that UDP_{NoPR} contributes with 4 UAS points on average over the baseline. Nevertheless, the baseline is oracle-informed about the language’s best branching direction, a property that UDP does not have. Instead, the decoding step determines head direction as described in Section 3.2. Complementarily, we can measure the contribution of PR by observing the difference between regular UDP and UDP_{NoPR} . The latter scores on average 9 UAS points lower than UDP. These 9 points are caused by the difference attachment of content words in the first decoding step.

6.2 Breakdown by POS

UD is a constantly improving effort, and not all v1.2 treebanks have the same level of formalism compliance. Thus, the interpretation of, e.g., the AUX-VERB or DET-PRON distinctions might differ across treebanks. However, we ignore these differences in our analysis and consider all treebanks to be equally compliant.

The root accuracy scores oscillate around an average of 69%, with Arabic and Tamil (26%) and Estonian (93%) as outliers. Given the PR personalization (Sec. 3.1), UDP has a strong bias for choosing the first verb as main predicate. Without personalization, performance drops 2% on average. This difference is consistent even for verb-final languages like Hindi, given that the main verb of a simple clause will be its only verb, regardless of where it appears. Moreover, using PR personalization makes the ranking calculations converge a whole order of magnitude faster.

The bigram heuristic to determine adposition direction succeeds at identifying the predominant pre- or postposition preference for all languages (average ADP UAS of 75%). The fixed direction for the other functional POS is largely effective, with few exceptions, e.g., DET is consistently right-attaching on all treebanks except Basque (average overall DET UAS of 84%, 32% for Basque). These alternations could also be estimated from the data in a manner similar to ADP. Our rules do not make nouns eligible heads for verbs. As a result, the system cannot infer relative clauses. We have excluded the NOUN \rightarrow VERB rule during development because it makes the hierarchy between verbs and nouns less conclusive.

We have not excluded punctuation from the evaluation. Indeed, the UAS for the PUNCT is low

(an average of 21%, standard deviation of 9.6), even lower than the otherwise problematic CONJ. Even though conjunctions are pervasive and identifying their scope is one of the usual challenges for parsers, the average UAS for CONJ is much larger (an average of 38%, standard deviation of 13.5) than for PUNCT. Both POS show large standard deviations, which indicates great variability. This variability can be caused by linguistic properties of the languages or evaluation datasets, but also by differences in annotation convention.

6.3 Cross-domain consistency

Models with fewer parameters are less likely to overfit for a certain dataset. In our case, a system with few, general rules is less likely to make attachment decisions that are very particular of a certain language or dataset. Plank and van Noord (2010) have shown that rule-based parsers can be more stable to domain shift. We explore if their finding holds for UDP as well, by testing on i) the UD development data as a readily available proxy for domain shift, and ii) manually curated domain splits of select UD test sets.

Language	Domain	BL_G	MSD_G	UDP_G	MSD_P	UDP_P
Bulgarian	bulletin	48.3	<u>67.5</u>	67.4	<u>65.4</u>	<u>61.5</u>
	legal	<u>47.9</u>	76.9	69.2	73.0	68.6
	literature	53.6	74.2	69.0	72.8	66.6
	news	49.3	74.6	70.2	73.0	68.2
	various	51.4	74.2	72.5	72.6	69.5
Croatian	news	<u>41.2</u>	<u>62.4</u>	57.9	61.8	<u>52.2</u>
	wiki	41.9	64.8	<u>55.8</u>	<u>58.2</u>	56.3
English	answers	44.1	61.6	55.9	59.5	53.7
	email	42.8	58.8	52.1	57.1	56.3
	newsgroup	<u>41.7</u>	55.5	<u>49.7</u>	52.9	51.1
	reviews	47.4	66.8	54.9	63.9	52.2
	weblog	43.3	<u>51.6</u>	50.9	<u>49.8</u>	53.8
	magazine†	41.4	60.9	55.6	58.4	53.3
	bible†	38.4	56.2	56.2	56.8	48.6
	questions†	38.7	69.7	55.6	60.5	<u>47.2</u>
Italian	europarl	50.8	<u>64.1</u>	70.6	<u>62.7</u>	69.7
	legal	51.1	67.9	69.0	<u>64.4</u>	67.2
	news	49.4	68.9	67.5	67.0	<u>65.3</u>
	questions	<u>48.7</u>	80.0	77.0	79.1	76.1
	various	49.7	67.8	69.0	65.3	67.6
	wiki	51.8	71.2	68.1	70.3	66.6
Serbian	news	42.8	<u>68.0</u>	58.8	65.6	<u>53.3</u>
	wiki	<u>42.4</u>	68.9	58.8	<u>62.8</u>	55.8

Table 6: Evaluation across domains. UAS for baseline with gold POS (BL_G), UDP with gold POS (UDP_G) and predicted POS (UDP_P), and multi-source delexicalized with gold and predicted POS (MSD_G and MSD_P). English datasets marked with † are in-house annotated. Lowest results per language underlined. Bold: UDP outperforms MSD.

Development sets We have used the English development data to choose which relations would be included as head rules in the final system (Table 1). It would be possible that some of the rules are indeed more befitting for the English data or for that particular section.

However, if we regard the results for UDP_G in Table 5, we can see that there are 24 languages (out of 32) for which the parser performs better than for English. This result indicates that the head rules are general enough to provide reasonable parses for languages other than the one chosen for development. If we run UDP_G on the development sections for the other languages, we find the results are very consistent. Any language scores on average ± 1 UAS with regards to the test section. There is no clear tendency for either section being easier to parse with UDP.

Cross-domain test sets To further assess the cross-domain robustness, we retrieved the domain (genre) splits from the test sections of the UD treebanks where the domain information is available as sentence metadata: from Bulgarian, Croatian, and Italian. We also include a UD-compliant Serbian dataset which is not included in the UD release but which is based on the same parallel corpus as Croatian and has the same domain splits (Agić and Ljubešić, 2015). When averaging we pool Croatian and Serbian together as they come from the same dataset.

For English, we have obtained the test data splits matching the sentences from the original distribution of the English Web Treebank. In addition to these already available datasets, we have annotated three different datasets to assess domain variation more extensively, namely the first 50 verses of the King James Bible, 50 sentences from a magazine, and 75 sentences from the test split in QuestionBank (Judge et al., 2006). We include the third dataset to evaluate strictly on questions, which we could do already in Italian. While the `answers` domain in English is made up of text from the Yahoo! Answers forum, only one fourth of the sentences are questions. Note these three small datasets are not included in the results on the canonical test sections in Table 5.

Table 7 summarizes the per-language average score and standard deviation, as well as the macro-averaged standard deviation across languages. UDP has a much lower standard deviation across domains compared to MSD. This holds across lan-

Language	BL_G	MSD_G	UDP_G	MSD_P	UDP_P
Bulgarian	50.1 \pm 2.4	73.5 \pm 3.5	69.7 \pm 1.8	71.3 \pm 3.3	66.9 \pm 3.2
Croatian+Serbian	42.1 \pm 0.7	66.0 \pm 3.0	57.8 \pm 1.4	62.1 \pm 3.0	54.4 \pm 2.0
English	42.2 \pm 2.8	60.1 \pm 6.2	53.9 \pm 2.5	57.3 \pm 4.3	52.0 \pm 3.3
Italian	50.3 \pm 1.2	70.0 \pm 5.4	70.1 \pm 3.3	68.1 \pm 6.0	68.7 \pm 3.9
<i>Average Std.</i>	1.8	4.5	2.5	4.2	3.1

Table 7: Average language-wise domain evaluation. We report average UAS and standard deviation per language. The bottom row provides the average standard deviation for each system.

guages. We attribute this higher stability to UDP being developed to satisfy a set of general properties of the UD syntactic formalism, instead of being a data-driven method more sensitive to sampling bias. This holds for both the gold-POS and predicted-POS setup. The differences in standard deviation are unsurprisingly smaller in the predicted POS setup. In general, the rule-based UDP is less sensitive to domain shifts than the data-driven MSD counterpart, confirming earlier findings (Plank and van Noord, 2010).

Table 6 gives the detailed scores per language and domain. From the scores we can see that `presidential bulletin`, `legal` and `weblogs` are amongst the hardest domains to parse. However, the systems often do not agree on which domain is hardest, with the exception of `Bulgarian bulletin`. Interestingly, for the Italian data and some of the hardest domains UDP outperforms MSD, confirming that it is a robust baseline.

6.4 Comparison to full supervision

In order to assess how much information the simple principles in UDP provide, we measure how many gold-annotated sentences are necessary to reach its performance, that is, after which size the treebank provides enough information for training that goes beyond the simple linguistic principles outlined in Section 3.

For this comparison we use a first-order non-projective TurboParser (Martins et al., 2013) following the setup of Agić et al. (2016). The supervised parsers require around 100 sentences to reach UDP-comparable performance, namely a mean of 300 sentences and a median of 100 sentences, with Bulgarian (3k), Czech (1k), and German (1.5k) as outliers. The difference between mean and median shows there is great variance, while UDP provides very constant results, also in terms of POS and domain variation.

7 Conclusion

We have presented UDP, an unsupervised dependency parser for Universal Dependencies (UD) that makes use of personalized PageRank and a small set of head-dependent rules. The parser requires no training data and estimates adposition direction directly from the input.

We achieve competitive performance on all but two UD languages, and even beat a multi-source delexicalized parser (MSD) on Hindi. We evaluated the parser on three POS setups and across domains. Our results show that UDP is less affected by deteriorating POS tags than MSD, and is more resilient to domain changes. Given how much of the overall dependency structure can be explained by this fairly system, we propose UDP as an additional UD parsing baseline. The parser, the in-house annotated test sets, and the domain data splits are made freely available.⁶

UD is a running project, and the guidelines are bound to evolve overtime. Indeed, the UD 2.0 guidelines have been recently released. UDP can be augmented with edge labeling for some deterministic labels like `case` or `det`. Some further constraints can be incorporated in UDP. Moreover, the parser makes no special treatment of multiword expression that would require a lexicon, coordinations or proper names. All these three kinds of structures have a flat tree where all words depend on the leftmost one. While coordination attachment is a classical problem in parsing and out of the scope of our work, a proper name sequence can be straightforwardly identified from the part-of-speech tags, and it falls thus in the area of structures predictable using simple heuristics. Moreover, our use of PageRank could be expanded to directly score the potential dependency edges instead of words, e.g., by means of edge reification.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Héctor Martínez Alonso is funded by the French DGA project VerDi. Barbara Plank thanks the Center for Information Technology of the University of Groningen for the HPC cluster. Željko Agić and Barbara Plank thank the Nvidia Corporation for supporting their research. Anders Søgaard is funded by the ERC Starting Grant LOWLANDS No. 313695.

⁶https://github.com/hectormartinez/ud_unsup_parser

References

- Steven Paul Abney. 1987. *The English noun phrase in its sentential aspect*. Ph.D. thesis, Massachusetts Institute of Technology.
- Željko Agić, Anders Johannsen, Barbara Plank, Héctor Alonso Martínez, Natalie Schluter, and Anders Søgaard. 2016. Multilingual Projection for Parsing Truly Low-Resource Languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.
- Željko Agić and Nikola Ljubešić. 2015. Universal Dependencies for Croatian (that Work for Serbian, too). In *Proceedings of the Sixth Workshop on Balto-Slavic Natural Language Processing*, pages 1–8, Hissar, Bulgaria.
- Željko Agić, Dirk Hovy, and Anders Søgaard. 2015. If All You Have is a Bit of the Bible: Learning POS Taggers for Truly Low-Resource Languages. In *Proceedings of ACL*, pages 268–272, Beijing, China.
- Emily M. Bender. 2009. Linguistically naïve != language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction Between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32, Athens, Greece.
- Thorsten Brants. 2000. TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231.
- Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117.
- Dipanjan Das and Slav Petrov. 2011. Unsupervised Part-of-Speech Tagging with Bilingual Graph-Based Projections. In *Proceedings of ACL*, pages 600–609, Portland, Oregon, USA.
- Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal Stanford Dependencies: A Cross-Linguistic Typology. In *Proceedings of LREC*, pages 4585–92, Reykjavik, Iceland.
- Douwe Gelling, Trevor Cohn, Phil Blunsom, and Joao Graça. 2012. The Pascal Challenge on Grammar Induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80, Montreal, Canada.
- Jennifer Gillenwater, Kuzman Ganchev, Joao Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in Dependency Grammar Induction. In *Proceedings of ACL*, pages 194–199, Uppsala, Sweden.

- Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping Parsers via Syntactic Projection Across Parallel Texts. *Natural Language Engineering*, 11(03):311–325.
- Anders Johannsen, Héctor Martínez Alonso, and Barbara Plank. 2015. Universal dependencies for Danish. In *Proceedings of The International Workshop on Treebanks and Linguistic Theories (TLT14)*, pages 157–167, Warsaw, Poland.
- Anders Johannsen, Željko Agić, and Anders Søgaard. 2016. Joint Part-of-Speech and Dependency Projection from Multiple Sources. In *Proceedings of ACL*.
- John Judge, Aoife Cahill, and Josef Van Genabith. 2006. Questionbank: Creating a Corpus of Parse-Annotated Questions. In *Proceedings of ACL*, pages 497–504, Sydney, Australia.
- Dan Klein and Christopher Manning. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of ACL*, pages 478–485, Barcelona, Spain.
- Peter Lofgren. 2015. *Efficient Algorithms for Personalized PageRank*. Ph.D. thesis, Stanford Computer Science.
- Xuezhe Ma and Fei Xia. 2014. Unsupervised Dependency Parsing with Transferring Distribution via Parallel Guidance and Entropy Regularization. In *Proceedings of ACL*, pages 1337–1348, Baltimore, Maryland, USA.
- André F. T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers. In *Proceedings of ACL*, pages 617–622, Sofia, Bulgaria.
- Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-Source Transfer of Delexicalized Dependency Parsers. In *Proceedings of EMNLP*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Ryan McDonald, Joakim Nivre, Yvonne Quirnbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of ACL*, pages 92–97, Sofia, Bulgaria.
- Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using Universal Linguistic Knowledge to Guide Grammar Induction. In *Proceedings of EMNLP*, pages 1234–1244, Cambridge, Massachusetts, USA.
- Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Măranduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cene Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uribe, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal Dependencies 1.2.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*, pages 1659–1666, Portorož, Slovenia.
- Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab.
- Barbara Plank and Gertjan van Noord. 2010. Grammar-Driven versus Data-Driven: Which Parsing System Is More Affected by Domain Shifts? In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, pages 25–33, Uppsala, Sweden.
- Mohammad Sadeh Rasooli and Michael Collins. 2015. Density-Driven Cross-Lingual Transfer of Dependency Parsers. In *Proceedings of EMNLP*, pages 328–338, Lisbon, Portugal.
- Valentin Spitzkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From Baby Steps to Leapfrog: How Less is More in Unsupervised Dependency Parsing. In *Proceedings of NAACL*, pages 751–759, Los Angeles, California, USA.
- Valentin Spitzkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher Manning. 2010b. Viterbi Training Improves Unsupervised Dependency Parsing. In *Proceedings of CoNLL*, pages 9–17, Uppsala, Sweden.

- Anders Søgaard. 2011. Data Point Selection for Cross-Language Adaptation of Dependency Parsers. In *Proceedings of ACL*, pages 682–686, Portland, Oregon, USA.
- Anders Søgaard. 2012a. Two Baselines for Unsupervised Dependency Parsing. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 81–83, Montreal, Canada.
- Anders Søgaard. 2012b. Unsupervised dependency parsing without training. *Natural Language Engineering*, 18(02):187–203.
- Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*. Editions Klincksieck.
- Jörg Tiedemann and Željko Agić. 2016. Synthetic Treebanking for Cross-Lingual Dependency Parsing. *Journal of Artificial Intelligence Research*, 55:209–248.
- Jörg Tiedemann. 2014. Rediscovering Annotation Projection for Cross-Lingual Parser Induction. In *Proceedings of COLING*, pages 1854–1864, Dublin, Ireland.
- David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing Multilingual Text Analysis Tools via Robust Projection Across Aligned Corpora. In *Proceedings of HLT*, pages 200–207, San Diego, California, USA.
- Daniel Zeman and Philip Resnik. 2008. Cross-Language Parser Adaptation Between Related Languages. In *Proceedings of IJCNLP*, pages 35–42, Hyderabad, India.

Delexicalized Word Embeddings for Cross-lingual Dependency Parsing

Mathieu Dehouck¹ and Pascal Denis²

¹Univ. Lille, CNRS, UMR 9189 - CRISAL

^{1,2}Magnet Team, Inria Lille - Nord Europe

59650 Villeneuve d'Ascq, France

{mathieu.dehouck, pascal.denis}@inria.fr

Abstract

This paper presents a new approach to the problem of cross-lingual dependency parsing, aiming at leveraging training data from different source languages to learn a parser in a target language. Specifically, this approach first constructs word vector representations that exploit structural (i.e., dependency-based) contexts but only considering the morpho-syntactic information associated with each word and its contexts. These delexicalized word embeddings, which can be trained on any set of languages and capture features shared across languages, are then used in combination with standard language-specific features to train a lexicalized parser in the target language. We evaluate our approach through experiments on a set of eight different languages that are part the Universal Dependencies Project. Our main results show that using such delexicalized embeddings, either trained in a monolingual or multilingual fashion, achieves significant improvements over monolingual baselines.

1 Introduction

Over the recent years, distributional and distributed representations of words have become a critical component of many NLP systems (Turian et al., 2010; Collobert et al., 2011). The reason for this success is that these low-dimensional, dense word vectors address two major problems that appear in many NLP applications, namely data sparsity and the inherent lack of expressivity of one-hot representations, and they can also be trained on large unannotated corpora which are cheap to produce and easily available. While they have proven

useful in a number of tasks, and especially in dependency parsing (Koo et al., 2008), these word vectors are often learned in a generic manner, only using linear bag-of-word contexts (F. Brown et al., 1992; Mikolov et al., 2013), without paying much attention to the specifics of the task to be solved. Only very recently, people have tried to learn dependency-based embeddings (Bansal et al., 2014; Levy and Goldberg, 2014; Madhyastha et al., 2014; Bansal, 2015), and these new structure-aware representations have been shown to improve parsing performance in a monolingual setting.

We would like to generalize this idea to a multilingual setting in a way that allows the transfer of structural information associated with words from one (or several) languages to others. While previous work have attempted to learn multilingual word clusters or embeddings (Guo et al., 2015) and use these for cross-lingual transfer, this paper explores a different research direction. Specifically, we investigate the use of vectorial representations in which lemma information have been abstracted away from both words and contexts, hence reduced to their morpho-syntactic attributes. The appeal of these de facto *delexicalized* word representations is that they further increase the coverage over the available training data, potentially allowing for better generalization. Furthermore, while words tend to be hard to align in a cross-lingual setting due to homonymy and polysemy, morpho-syntactic information tends to be much more robust to language barrier (depending on typological closeness), which make them particularly relevant for cross-lingual transfer. In contrast with delexicalized parsing approaches (McDonald et al., 2011), the proposed method uses delexicalization during word embedding learning, not during parsing. Once induced over different source language datasets, these language-shared

representations are used as additional features, in combination with standard language-specific features, in a standard lexicalized monolingual graph-based dependency parser for the target language. As far as we know, this is the first attempt at constructing delexicalized word embeddings for cross-lingual dependency parsing.

Through the use of these delexicalized word embeddings, we wish to explore two main hypotheses. First and foremost, we want to assess to what extent a parser for a particular language can benefit from using morpho-syntactic regularities extracted from other languages. By comparing the performance of embeddings learned on different sets of source languages in parsing a specific target language, we also hope to assess whether and which typological similarities impact the learning of “good” embeddings. The second hypothesis looks at the choice of context types (sequential vs structural) used for learning these embeddings. That is, we wish to see if the use of syntactic structure delivers better parsing improvements, again in relation to typological similarities or differences. These hypotheses will be tested on treebanks from the recent Universal Dependencies Project (Nivre et al., 2016), which provides us with homogeneous syntactic dependency annotations in many languages.

In section 2, we provide some background and review previous work on graph-based dependency parsing for mono- and cross-lingual settings and on word embeddings. Section 3 details our approach for learning delexicalized word embeddings and using them in dependency parsing. In section 4, we present some experimental results that show the importance of grammatical embeddings for the task at hand. And finally in section 5, we draw some conclusions and present future perspectives.

2 Preliminaries and Related Work

The approach proposed in this paper draws on three different lines of work in dependency parsing.

2.1 Graph-based Dependency Parsing

A dependency tree is a graphical representation of the syntactic structure of a sentence. The task of dependency parsing is to predict the dependency tree of a given sentence x , \mathcal{T}_x being the set of all its possible trees. Assuming we have access to a

scoring function $Score(\bullet, \bullet)$ that tells how well a dependency tree fits the syntactic structure of a sentence, the goal of dependency parsing is to find the tree \hat{y} such that:

$$\hat{y} = \operatorname{argmax}_{t \in \mathcal{T}_x} Score(x, t).$$

The size of \mathcal{T}_x grows exponentially with the length of x , $|\mathcal{T}_x| = |x|^{|x|-2}$, making an exhaustive search for the best tree impractical in most cases. Thus in practice, some simplifying assumptions are made. Here we use the graph-based, edge-factored approach based on the assumption that the score of a tree can be computed as the sum of its edges scores (McDonald et al., 2005a). Let $score(\bullet, \bullet)$ be a scoring function for edges.

$$Score(x, t) = \sum_{e \in t} score(x, e).$$

In this case, finding the best parse tree for x boils down to finding the maximum spanning tree in the complete graph whose vertices are the words of x . The score of an edge e is here defined as the dot product between a model w (a weight vector) and the feature vector $\phi(x, e)$ of this edge.

$$score(x, e) = w \cdot \phi(x, e).$$

In this paper, we learn the model w in an online manner with the Passive-Aggressive (PA) algorithm described in (Crammer et al., 2006). Specifically, we use the PA-II that uses a squared hinge loss in its predicted-loss cost-sensitive version, in which the cost is computed in terms of the symmetric difference on the edges with respect to the target tree.

2.2 Word Vectors for Dependency Parsing

Distributional and distributed word representations are dense vectorial representations of words that live in a multi-dimensional integer or real space whose size is much smaller than the size of the original language vocabulary. There are now a large variety of spectral and neural approaches for learning these representations, including several variants of Principal Component Analysis (Jolliffe, 2002) and several deep neural net approaches. Most of these approaches have in common that they solely exploit linear, bag-of-word co-occurrence between words to derive these low-dimensional representations (Mikolov et al., 2013; Lebret and Collobert, 2014).

Starting with the work of Koo (2008), the inclusion of this type of low-dimensional word representations as features has been shown to be a

simple yet very effective way of improving dependency parsing performance. The main appeal of these low-dimensional dense representations is that they mitigate major shortcomings of standard one-hot encoding representations which are very sparse and live in very high dimensional spaces, thus lacking in expressivity and hindering generalization. While it is still unclear whether pre-trained embeddings (Andreas and Klein, 2014) indeed capture interesting syntactic information, more recent work have concentrated on learning dependency-based word embeddings (Bansal et al., 2014; Levy and Goldberg, 2014; Madhyastha et al., 2014; Bansal, 2015). In these approaches, word co-occurrences are defined in terms of dependency contexts (x is the governor of word w), instead of linear contexts (x appears within a range of s around word w). Embedding techniques have also started to be applied to objects other than words, namely on dependency relations (Bansal, 2015; Kiperwasser and Goldberg, 2015).

In this paper, we depart from these approaches by learning a low-dimensional word vector representation that is based only on the morpho-syntactic information associated with that word, and learning is performed with simple PCA. Furthermore, we do not use any auto-parsed data in order to avoid errors from spreading into the embedding. Another point is that even if we use a first order parsing model, we use higher-order contexts for learning the embeddings. Bansal (2015) also uses higher-order contexts but combines them with a second-order dependency model.

2.3 Cross-lingual Dependency Parsing

Cross-lingual dependency parsing encompasses several problems ranging from learning a parser for a target language relying solely on annotated data from a source language (Lynn et al., 2014) to learning a unique parser that can handle various languages (Ammar et al., 2016). Delexicalized parsers (McDonald et al., 2011) have been used to avoid the problems that arise from lexical translation. More recently, cross-lingual parsers have been trained using cross-lingual word clusters as well as multilingual word embeddings (Guo et al., 2015) to alleviate the lack of lexical information.

Our work differs from previous studies in that it assumes the availability of annotated data from the target language for training the parser, but uses multilingual embeddings to benefit from annotated

data in other languages. Another important difference is that while multilingual word embeddings are usually used to replace lexical items, we use morpho-syntactic embeddings that are less language dependent.

3 Dependency Parsing with Delexicalized Word Embeddings

Standard word embeddings have two major drawbacks for our purposes: they represent word forms which are not easy to transfer from one language to another, and they rely on sequential contexts which are not grammatically motivated for languages with free word order.

To circumvent these problems, we propose to create embeddings for morpho-syntactic attribute sets using structural information from dependency trees. As we abstract away the lexical form of words we call our embeddings *delexicalized word embeddings*. The advantage of embedding sets of morpho-syntactic attributes over word forms is that morpho-syntactic attributes are shared across languages much more frequently than lexical features, and they also tend to be more stable through translation. This allows a more reliable transfer of linguistic knowledge from one language to another. This also increases the vocabulary coverage as the number of morpho-syntactic attributes is far smaller than the number of word forms. Here, we choose to learn representations for *full* attribute sets (i.e., the set containing all the morpho-syntactic attributes associated with a word form) instead of learning representations for single attributes and then composing those for each word. This is in line with standard word embedding approaches which implicitly do the same in learning a different representation for each distinct word form of a lemma (e.g., *be*, *am*, *is*, *were*) without any further analysis. We discuss this issue in more detail in the experiment section.

3.1 Delexicalized Words

Let us briefly illustrate the kinds of morpho-syntactic attributes we want to embed with some examples. For these examples, we are using the notation of the Universal Dependencies project.

The English word *a* is a *determiner* (its part-of-speech or POS is DET), its number is *singular* and its definiteness is *indefinite*. As a determiner, it does not have tense or mood, and as most English words, it does not have gen-

der. We would thus embed the feature set $DET:Definite=Ind|Number=Sing$.

Looking at an example from a morphologically richer language, we have Finnish verb form *oli* (meaning *was*) which we encode as $VERB:Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin|Voice=Act$. This means that *oli* is a finite verb form (not a participle) in the indicative past, its voice is active and its agreement is third person, singular.

This gives vocabulary sizes ranging from a hundred or less for analytic languages (English has about 120 such combinations) to several thousands for synthetic ones (the Finnish part of the UD Project has about 4000 such combinations).

3.2 Structural Contexts

Having decided to embed morpho-syntactic attribute sets rather than words, we need to map these new objects to dense vectors in \mathbb{R}^d . To this end, we apply principal component analysis (PCA) to a co-occurrence matrix (possibly re-weighted) whose lines represent our new attribute sets (i.e., our delexicalized words) and whose columns are their contexts, which are also expressed in terms of morpho-syntactic attributes. We describe those contexts later on, but it is worth noting that as we allow them to diverge from the morpho-syntactic features sets, not including every features for example, the number of context can potentially reach a few tens of thousands.

As we want to learn embeddings specifically tailored to dependency parsing, we use not only sequential (i.e. linear) contexts but also structural contexts based on dependency trees. Sequential contexts are of the kind: “word x appears in a window of size l centered on word y ”. Structural contexts are instead defined on the dependency tree: “word x is the governor of word y ”, or “ x is a dependent of y ” or again “ x is a sibling of y ”.

The new structural contexts can be seen as following a certain path in the dependency tree linking two words. Let $up(x, t)$ be the function that maps a word x to its governor (following the *upgoing* edge) in tree t . As the root of a tree has no governor we add a dummy token called *nil* for that purpose. Then $down(x, t)$ is the function that maps x to the set of its dependents in t .

$$down(x, t) = \{y \in t \mid up(y, t) = x\}$$

Then we can define our new contexts as combinations of *up* and *down*, for example the governor

of x is $up(x, t)$, its dependents are $down(x, t)$ and its siblings are $down(up(x, t), t) \setminus \{x\}$.

We can also define similar functions over sequences. Let $right(x, s)$ (respectively $left(x, s)$) be the word in sequence s standing just at the right (respectively at the left) of x , then we can also express the sequential contexts in the same framework. We also add two new dummy tokens *begin* and *end* to avoid ill definition at the borders of s . Using the notation $f^n(\bullet)$ for $f \circ f \circ \dots \circ f(\bullet)$ where f is applied n times, we have that the window of size l centered on x is $\{right^i(x, s) \mid i \in [i..l]\} \cup \{left^i(x, s) \mid i \in [i..l]\}$. We can also define new contexts such as left or right siblings.

Let us now turn to an example to make our approach more concrete. Figure 1 shows a dependency tree for a Gothic sentence¹ from the Universal Dependencies Project. Each word is accompanied with its part-of-speech and the corresponding morpho-syntactic attributes. Colored links represent examples of contexts, orange links standing for contexts of length 1 and blue links standing for contexts of length 2.

Embedding Delexicalized Words With Structural Contexts

Given the above definition of structural contexts, there are still several design parameters to set in order to construct embeddings. First, we distinguish different *types of contexts*, depending on whether the contexts are sequential (with a distinction between left and right), governor, dependents and siblings (with a distinction between first left sibling, first right sibling and others). Second, there are different *context spans*, where the span is the maximum length (in term of function applications) of a context. It is equivalent to the window size for sequential context. For example, the sibling context has a fixed span of 2, but the governor of span 2 means the governor of the governor (if it exists, *nil* otherwise). Third, we distinguish different *granularity levels* of contexts, corresponding to the maximum number of morpho-syntactic attributes used to model contexts. Like words, contexts are also defined in terms of morpho-syntactic attributes, but we do not require complete sets for them, hence allowing for different granularity

¹The reader may notice that there is no punctuation in that sentence, it is because there is originally no punctuation in Ulfila’s Bible from which the sentence comes, but if there were some (as in modern texts) we would use them just as usual, treating any token as a word.

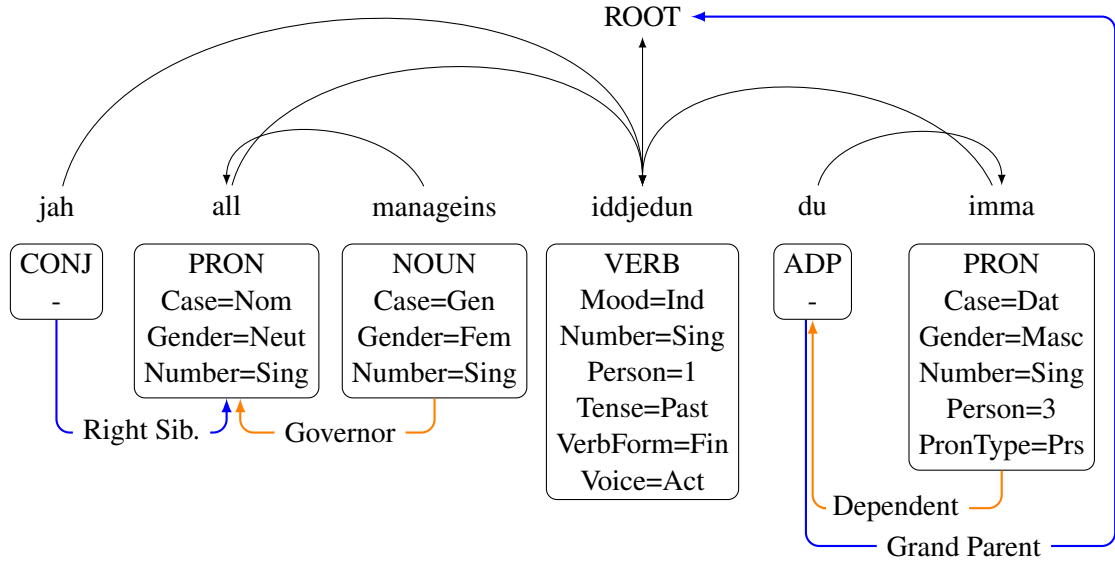


Figure 1: Dependency tree of sentence *jah all manageins iddjedun du imma* (and all of the crowd went to him) from Ulfila’s Bible, from the Gothic part of the Universal Dependencies Project. Under the words are their morpho-syntactic analysis and the colored links represent some possible structural contexts.

levels. Taking an example from the sentence of Figure 1, with a granularity of 0, the word *manageins* triggers the context *NOUN* only (i.e., the POS tag alone). With a granularity of 1, it triggers *NOUN:Case=Gen*, *NOUN:Gender=Fem* and *NOUN:Number=Sing* (i.e., the word POS tag is crossed with each of the other attributes). And with *full* granularity, the union over all subsets of attributes (of size 0 to 3) is combined with the POS tag. This gives 2^a possible contexts for a morpho-syntactic attributes set with a attributes.

Finally, other parameters are the embedding algorithm (here we use PCA), the matrix normalization method (here we use a simple L2-norm row normalization), the size of the embedding space and the number of contexts used. We can keep all the contexts as their numbers range from 36 (each part-of-speech counted twice for before and after a word and two extra context representing the beginning and the end of the sentence) to a few tens of thousands.

3.3 From Word Embeddings to Edge Embeddings to Parsing Features

As the dependency model factors on edges, we need to turn the word embeddings into edge embeddings. We want an aggregating function that preserves edge orientation and the dependency between the edge endpoints. So we chose to use the

outer product of the two original embeddings because it is not commutative and each output dimension depends on the two inputs. Let \otimes denote the outer product and let u and v be two vectors of \mathbb{R}^d , then:

$$u \otimes v = uv^\top$$

This operation yields a matrix in $\mathbb{R}^{d \times d}$ but we need a vector, so we take the vector $vec(uv^\top)$. In the following, whenever we use a matrix where a vector is expected, we implicitly assume the presence of a $vec(\bullet)$ ². There is a slight scalability problem as the output size grows quadratically with the size of the inputs. But in the case of dependency parsing, where feature vectors are commonly a few millions dimensions long, for typical embeddings size (between 100 and 500) an increase of a few tens of thousands dimensions is acceptable.

We would like to use more context than just the two nodes of the edge to represent it. In order to define higher-order contexts, we use triplets of delexicalized words centered on each side of the edge. Specifically, we concatenate the representations associated with the triplets to keep a tractable model of size $9d^2$. Note that applying an outer product across each word of the triplet would be

²In this case, it does not matter if a matrix is vectorized by the columns or by the rows, as soon as the same convention is used consistently throughout the algorithm.

prohibitive, leading to vectors of the order of 10^{12} dimensions which would not even fit in memory.

More formally, let \oplus denote the concatenation operator, $Dep(e)$ (respectively $Gov(e)$) be the dependent (respectively the governor) of an edge e and let $emb(\bullet)$ be the embedding function. $\phi_{emb}(\bullet, \bullet)$ being the embedding part of the feature vector. Using the notation of section 2.1, we have:

$$\begin{aligned} \phi_{emb}(x, e) = & \bigoplus_{i \in \{-1, 0, +1\}} emb(Gov(e)_i) \\ & \otimes \bigoplus_{i \in \{-1, 0, +1\}} emb(Dep(e)_i). \end{aligned}$$

$$score(x, e) = \mathbf{w} \cdot (\phi(x, e) \oplus \alpha \phi_{emb}(x, e)).$$

Where α is a scalar allowing to tune the relative importance of each part of the feature vector, and $\phi(\bullet, \bullet)$ is the traditional dependency feature vector. This follows the same approach as Kiperwasser and Goldberg (2015) and Chen et al.(2014). As mentioned previously, we have special tokens such as *begin* and *end* to represent the word before the beginning and the end of a sentence. We also have a *root* token that stands for the extra root node added by the graphical dependency model. And we also have a back-off embedding of raw part-of-speech to handle unseen delexicalized words in the test set.

4 Experiments

4.1 Data set

We have tested our parsing models based on delexicalized word embeddings on eight languages using the data of the Universal Dependencies (UD) v1.3 (Nivre et al., 2016). We have chosen to work on English (en), Basque (eu), Finnish (fi), French (fr), Gothic (got), Hebrew (he), Hungarian (hu) and Romanian (ro). These languages belong to four different families, which are Indo-European (en, fr, got, ro), Finno-Ugric (fi, hu), Semitic (he), and Basque which forms a separate group. They also display various levels of morphological complexity not correlated with the families (en, fr and he do not have case marking while the other five do to various degrees) as well as different grammatical typologies (eu is an ergative language, while the other seven are accusative ones). When several corpora are available for a language, we picked the standard one. Table 1 provides some basic statistics on the language datasets. Also note that our experiments follow the train/dev/test split as pro-

vided by the UD Project.

4.2 Features

Dependency Features

For parsing, we use standard graphical dependency parsing features that include word forms and POS-tags of edge nodes and surrounding words, edge length and direction and conjunction of those basic features. The main difference with the original MSTparser features of McDonald et al.(2005b) is that instead of using truncated words of length 5 as back-off features, we use the lemmas that are provided in the UD Project.

Embedding Contexts

For the embedding contexts, we consider four parameters, namely the type and span of contexts, the granularity of the morpho-syntactic attributes used in those contexts and the dimension of the embedding space. Regarding the type of contexts we have experiments with three settings: (i) strictly sequential contexts (*Seq*), (ii) strictly structural contexts that use governor, dependents and siblings information (*Struct*) and (iii) mixed contexts using both dependency-based and sequential contexts (*Mix*). Regarding the span, we have tried 1 and 2. Siblings are only used in structural and mixed contexts of span 2 because that is the length of the path between a vertex and its siblings in a tree. We have tried granularity of 0 and full granularity. For the embedding space dimension we have tried 50, 150 and 500 dimensions, or the maximum possible size³ if smaller than 500. For better readability, we will use shortcuts to refer to the different parameter settings: 1 = (span 1, granularity 0), 2 = (span 2, granularity 0) and 3 = (span 2, full granularity for contexts span 1, granularity 0 for context span 2).

4.3 Experimental Settings

We have carried out two sets of experiments: monolingual and cross-lingual. In the first set, embeddings are learned on a single language training set and then used to parse that same language. In the second set, we have defined several clusters of languages based on their phylogenetic relationship and typological similarities. For a given cluster, embeddings are learned on the training sets of

³Spectral-based dimension reduction such as PCA are limited by the number of eigenvectors of the matrix to be reduced. For example a matrix of size 200×500 can at most be reduced into a 200×200 matrix via PCA. When the number of eigenvectors is smaller than 500, we use that value instead.

	Train		Test		morpho-syntactic tokens	POS
	sentences	words	sentences	words		
English	12 543	204 586	2 077	25 096	118	17
Basque	5 396	72 974	1 799	24 374	845	16
Finnish	12 217	162 721	648	9 140	1 592	15
French	14 557	356 216	298	7 018	195	17
Gothic	4 360	44 722	485	5 158	662	13
Hebrew	5 142	15 558	491	12 125	480	16
Hungarian	1 433	23 020	188	4 235	651	16
Romanian	4 759	108 618	794	18 375	412	17

Table 1: Number of sentences and words in the training and test sets, number of delexicalized word and of POS-tags for each language. The total number of embedded tokens is $|\text{morpho-syntactic feature set}| + |\text{POS}| + 3$ because of the POS back-offs and the special *begin*, *end* and *root* tokens.

each language in that cluster, and in turn used to parse each language in that cluster. It is possible not to use any data from the target language when learning the embeddings, but in this study we stick to using target language data.

Besides embeddings, there are three additional hyper-parameters that need to be tuned: the C aggressiveness parameter of the PA-II algorithm, the scaling factor α that controls the relative weight of the embedding features in the edge scores, and the number i of training iterations of the PA-II algorithm. We have tuned these hyper-parameters through a grid search on the development sets and picked the values that were behaving best on average, giving $C = 0.001$, $\alpha = 0.001$, $i = 5$.

All the scores reported below are Unlabeled Attachment Scores (UAS) measured on the test sets ignoring the punctuation marks. As a baseline comparison we use our implementation of the MSTparser without morpho-syntactic attributes representation of any kind. We computed the significance of the scores using the McNemar’s test.

4.4 Monolingual Experiments

Table 2 displays UAS scores for the monolingual setting. Except for French and Romanian that do not show real improvement, the six other languages show substantial performance increases with the embeddings. These improvements are statistically significant for all languages, except for Basque and Hebrew. One of our hypotheses was that structure is important when learning an embedding for dependency parsing and indeed our results support it. The largest improvements appear with structural or mixed embeddings which rely on syntactic structures.

The results for English are significant and close to each other for all types of embeddings, this tends to show that in English, sentence structure and word order are very correlated and both contribute information. Indeed that is what one expects for English which has a rigid syntax and a poor morphology.

On the other side of the picture, Basque and Gothic display the largest improvements with structural morpho-syntactic embeddings. This is also expected as those are both morphologically rich languages with more flexible word order. Even though the argument is less clear for Hungarian and Finnish, they both show that structure is important for learning informative dependency embeddings.

4.5 Cross-lingual Experiments

Table 3 summarizes the UAS scores achieved using delexicalized embeddings learned on several languages. Parsing accuracy improve for four languages (en, eu, hu, ro) in the cross-lingual setting compared to the best monolingual setting. While the multilingual embeddings do not outperform the monolingual ones for the other four languages, they still deliver parsing performance that are better than with the baseline MST parser for all languages (but Gothic). That shows that indeed using data from other languages is beneficial for learning good embeddings for dependency parsing, which was the second hypothesis we wanted to evaluate. We also notice that the largest gains are achieved with structural (or mixed) embeddings, giving more evidence to the importance of structure for learning embeddings for dependency parsing.

Language	Baseline	Seq 1	Seq 2	Seq 3	Struct 1	Struct 2	Struct 3	Mix 1	Mix 2	Mix 3
English	85.62	86.07 [*] ₄₀	85.92 [*] ₈₀	86.06 [◇] ₁₂₁	86.01 [◇] ₃₇	85.95 [*] ₁₂₁	85.94 [*] ₅₀	86.10 [*] ₇₇	86.19[*] ₅₀	85.84 ₁₂₁
Basque	76.65	76.60 ₃₈	76.70 ₇₆	76.73 ₅₀₀	76.67 ₃₅	76.85 ₁₁₉	76.90 ₁₅₀	76.66 ₅₀	76.72 ₁₅₀	76.80 ₅₀₀
Finnish	79.97	80.44 ₃₆	80.44 ₇₂	80.71 [◇] ₁₅₀	80.58 [*] ₃₃	80.35 ₁₁₄	80.58 [*] ₅₀	80.92[*] ₆₉	80.40 ₅₀	80.60 [*] ₅₀
French	83.99	83.48 ₄₀	83.55 ₅₀	83.47 ₁₅₀	83.42 ₃₇	83.68 ₁₂₈	83.71 ₁₅₀	83.61 ₇₇	83.89 ₁₅₀	83.84 ₁₉₈
Gothic	79.16	78.95 ₃₂	79.10 ₅₀	79.57 ₅₀₀	79.24 ₂₈	79.31 ₅₀	80.09[◇] ₅₀₀	79.59 ₅₀	79.62 ₁₅₀	79.62 ₅₀₀
Hebrew	84.05	83.87 ₃₈	83.86 ₅₀	84.24 ₅₀	84.18 ₃₅	84.32 ₅₀	84.14 ₁₅₀	84.32 ₅₀	84.34 ₅₀	84.36 ₁₅₀
Hungarian	79.15	79.23 ₃₈	80.13[*] ₇₆	79.83 [*] ₅₀₀	79.67 ₃₄	80.13[◇] ₁₁₈	79.69 ₅₀₀	79.94 [*] ₅₀	79.64 ₅₀	79.80 ₅₀
Romanian	81.35	81.34 ₄₀	81.29 ₈₀	81.21 ₄₁₅	81.00 ₃₇	81.38 ₅₀	81.29 ₁₅₀	81.30 ₅₀	81.26 ₁₅₀	81.21 ₄₁₅

Table 2: Best UAS scores for each embedding type in monolingual setting. The best score for each language are in bold and in gray are the results above the baseline. The statistical significance (using McNemar’s test) of an improvement over the baseline is indicated with a superscript mark: * stands for a significance with a p-value inferior than 0.05, \diamond stands for $p \leq 0.01$ and \diamond^* for $p \leq 0.001$. The length of the embeddings is reported as a subscript.

Language	Baseline	Best Mono	Best All	Best Multilingual
English	85.62	86.19 [*]	86.18 [*] _{seq3,50}	86.32[*] _{en,got,mix3,50}
Basque	76.65	76.90	76.97[*] _{struct3,50}	76.68 _{decl,seq2,50}
Finnish	79.97	80.92[*]	80.89 [*] _{struct2,50}	80.81 [◇] _{decl,seq2,50}
French	83.99	83.89	83.87 _{struct1,37}	83.89 _{en,fr,ro,mix1,77}
Gothic	79.16	80.09[◇]	79.80 _{struct2,50}	79.99 [*] _{got,ro,mix3,500}
Hebrew	84.05	84.36	84.32 _{seq3,150}	84.13 _{en,fr,he,mix1,77}
Hungarian	79.15	80.13 [*]	80.05 [*] _{mix2,150}	80.30[◇] _{decl,struct1,37}
Romanian	81.35	81.38	81.31 _{seq3,150}	81.52 _{hu,ro,mix3,50}

Table 3: Best UAS scores in cross-lingual setting. Under *Best All* are the results using the embeddings learned on the set of all languages, while under *Best Multilingual* are given the best results for each language using only a subset of the languages for learning the embedding. The subscript represents the context types and the number of dimensions of the embedding. The baselines and best monolingual scores are also reported. Significance of scores uses the same conventions as in Table 2.

Let us now look more closely at which groups of source languages are most helpful for specific target languages. First, note in general the best performing embeddings are never those obtained by using the full set of languages (this is only the case for Basque). This is expected since we have picked languages with very different grammars thus the full embeddings can be very noisy with regard to a single language. In fact, the Basque results are rather surprising since this language differs the most from the others in terms of morphology, but also one for which we had rather small training data.

The best parsing performance for English are achieved when using additional data from Gothic. As both are Germanic languages, this tends to show that data from genetically related languages

can help in learning a better representation. Even though they do not achieve the best results, similar patterns occur for French (French and Romanian are Romance languages and English has been heavily influenced by Romance languages) and for Gothic (Gothic and Romanian are both Indo-European languages). Similarly, Hungarian and Romanian reach their best scores when parsed with typologically close languages that have case marking. And again, Basque, Finnish and Gothic display similar patterns. Hebrew performs reasonably well with French and English which are two languages with fairly restricted word orders.

As to why some languages have better monolingual parsing results than multilingual results, we think this is at least partly due to the lack of flexibility of our model. That is, morpho-syntactic

attributes sets are treated independently from one another making some of them hard to use in the cross-lingual setting. For example, Hebrew verbs display ‘binyanim’ (internal flection classes) that do not appear in any other language, similarly Finnish has a lot of cases that are not found in other languages. Those are indeed two languages that do not perform well with other languages. We thus believe that introducing compositionality in our embedding model should help in solving those problems and enhance the results further.

5 Conclusion

In this paper, we have described a new way to induce multilingual embeddings, namely delexicalized word embeddings, that solely rely on the morpho-syntactic attributes of words which can easily be transferred across languages. This new approach to multilingual embeddings allows one to use annotated data from other languages to further improve the resulting embeddings and parsers, avoiding the problems that arise from lexicon alignment or cross-lingual word clustering.

In line with previous recent work, we have shown that the syntactic structure is crucial when it comes to learning embeddings for dependency parsing. In addition, we have seen that the impact of the structure on the quality of an embedding depends on language typology.

In future work, we should see how those morpho-syntactic embeddings can help in labeled dependency parsing, as edge types and word morpho-syntactic attributes are related. We would like to investigate the impact of the embedding algorithms (here we use PCA) on the final embeddings. We would also like to try other ways to turn word embeddings into edge embeddings in order to benefit more from the local neighborhoods. Finally, we would like to work on the embedding of clusters of morpho-syntactic attributes to induce higher-order embeddings for noun-phrases or verb-phrases and to deal with agreement and morpho-syntactic attributes hierarchy.

6 Acknowledgment

We thank the three anonymous reviewers for their comments. This work was supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

- Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. One parser, many languages. *CoRR*, abs/1602.01595.
- Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 822–827. Association for Computational Linguistics.
- Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics.
- Mohit Bansal. 2015. Dependency link embeddings: Continuous representations of syntactic substructures. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 102–108, Denver, Colorado, June. Association for Computational Linguistics.
- Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 816–826. Dublin City University and Association for Computational Linguistics.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. On-line passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December.
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, T. J. Watson, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics, Volume 18, Number 4, December 1992*.
- Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244. Association for Computational Linguistics.
- Ian T. Jolliffe. 2002. *Principal component analysis*. Springer series in statistics. Springer, New York, Berlin, Heidelberg. Autre(s) tirage(s) : 2004.

- Eliyahu Kiperwasser and Yoav Goldberg. 2015. Semi-supervised dependency parsing using bilexical contextual features from auto-parsed data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1348–1353. Association for Computational Linguistics.
- Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603. Association for Computational Linguistics.
- Rémi Lebreton and Ronan Collobert. 2014. Word embeddings through hellinger pca. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490. Association for Computational Linguistics.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308. Association for Computational Linguistics.
- Teresa Lynn, Jennifer Foster, Mark Dras, and Lamia Tounsi. 2014. Cross-lingual transfer parsing for low-resourced languages: An irish case study. In *Proceedings of the First Celtic Language Technology Workshop*, pages 41–49, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Swaroop Pranava Madhyastha, Xavier Carreras, and Ariadna Quattoni. 2014. Learning task-specific bilexical embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 161–171. Dublin City University and Association for Computational Linguistics.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 91–98. Association for Computational Linguistics.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005b. Spanning Tree Methods for Discriminative Training of Dependency Parsers.
- Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).
- Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Stance Classification of Context-Dependent Claims

Roy Bar-Haim¹, Indrajit Bhattacharya², Francesco Dinuzzo^{3*}
Amrita Saha², and Noam Slonim¹

¹IBM Research - Haifa, Mount Carmel, Haifa, 31905, Israel

²IBM Research - Bangalore, India

³IBM Research - Ireland, Damastown Industrial Estate, Dublin 15, Ireland

{roybar, noams}@il.ibm.com, {indrajitb, amrsaha4}@in.ibm.com

Abstract

Recent work has addressed the problem of detecting relevant claims for a given controversial topic. We introduce the complementary task of *claim stance classification*, along with the first benchmark dataset for this task. We decompose this problem into: (a) open-domain target identification for topic and claim (b) sentiment classification for each target, and (c) open-domain contrast detection between the topic and the claim targets. Manual annotation of the dataset confirms the applicability and validity of our model. We describe an implementation of our model, focusing on a novel algorithm for contrast detection. Our approach achieves promising results, and is shown to outperform several baselines, which represent the common practice of applying a single, monolithic classifier for stance classification.

1 Introduction

The need for making persuasive arguments arises in many domains, including politics, law, marketing, and financial and business advising. On-demand generation of pro and con arguments for a given controversial topic would therefore be of great practical value. Natural use cases include *debating support*, where the user is presented with persuasive arguments for a topic of interest, and *decision support*, where the pros and cons of a given proposal are presented to the user.

A notable research effort in this area is the IBM Debater® project whose goal is “to develop technologies that can assist humans to debate and

reason”¹. As part of this research, Levy et al. (2014) have developed *context-dependent claim detection*. Given a controversial *topic*, such as

- (1) *The sale of violent video games to minors should be banned*,

their system extracts, from corpora such as Wikipedia, *Context-Dependent Claims (CDCs)*, defined as “general, concise statements that directly support or contest the given Topic”. A claim forms the basis of an argument, being the assertion that the argument aims to establish, and therefore claim detection may be viewed as a first step in automated argument construction. Recent research on claim detection (Levy et al., 2014; Lippi and Torroni, 2015) was facilitated by the IBM argumentative structure dataset (Aharoni et al., 2014), which contains manually collected claims for a variety of topics, as well as supporting evidence.

In this work we introduce the related task of *Claim Stance Classification*: given a topic, and a set of claims extracted for it, determine for each claim whether it supports or contests the topic. Sorting extracted claims into *Pro* and *Con* would clearly improve the usability of both debating and decision support systems. We introduce the first benchmark for this task, by adding Pro/Con annotations to the claims in the IBM dataset.

Based on the analysis of this dataset, we propose a semantic model for predicting claim stance. We observed that both the debate topic and a supporting/contesting claim often contain a *target* phrase, about which they make a positive or a negative statement. The pro/con relation can then be determined by the sentiments of the topic and the claim towards their targets, as well as the semantic relation between these targets. For example, suppose that a topic expresses support for *freedom of*

¹http://researcher.ibm.com/researcher/view_group.php?id=5443

*Present affiliation - Amazon.

speech. A *Pro* claim may support it by arguing in favor of *free discussion*, or alternatively by criticizing *censorship*. We say that *freedom of speech* and *free discussion* are *consistent* targets, while *freedom of speech* and *censorship* are *contrastive*. Accordingly, we suggest that claim stance classification can be reduced to simpler, more tractable sub-problems:

1. Identify the targets of the given topic and claim.
2. Identify the polarity (*sentiment*) towards each of the targets.
3. Determine whether the targets are consistent or contrastive.

While our model seems intuitive, it was not clear a priori how well it captures the semantics of claims in practice. Some types of claims do not fit into this decomposition. Consider the following *Con* claim for the topic given in (1):

- (2) *Parents, not government bureaucrats, have the right to decide what is appropriate for their children.*

In this example, there is no clear sentiment target in the claim that is either consistent or contrastive with *the sale of violent video games to minors*. Nevertheless, extensive data annotation confirmed that our model is applicable to about 95% of the claims in the dataset, and for these claims, Pro/Con relations can be accurately predicted by solving the above sub-problems. Furthermore, our analysis reveals that *contrastive targets* are quite common, and thus must be accounted for. Our model highlights intriguing sub-problems such as *open-domain target identification* and *open-domain contrast detection* between a given pair of phrases, which have received relatively little attention in previous stance classification work. We hope that the annotated data collected in this work will facilitate further research on these important subtasks.

We developed a classifier for each of the above subtasks. Most notably, we present a novel method for the challenging task of contrast detection. Empirical evaluation confirms that our modular approach outperforms several strong baselines that employ a single, monolithic classifier.

2 Related Work

Previous work on *stance classification* focused on analyzing debating forums (Somasundaran and

Wiebe, 2009; Somasundaran and Wiebe, 2010; Walker et al., 2012b; Hasan and Ng, 2013; Walker et al., 2012a; Sridhar et al., 2014), congressional floor debates (Thomas et al., 2006; Yessenalina et al., 2010; Burfoot et al., 2011), public comments on proposed regulations (Kwon et al., 2007), and student essays (Faulkner, 2014). Most of these works relied on both generic features such as sentiment, and topic-specific features learned from labeled data for a closed set of topics. Simple classifiers with unigram or ngram features are known to be hard to beat for these tasks (Somasundaran and Wiebe, 2010; Hasan and Ng, 2013; Mohammad et al., 2016).

In addition to content-based features, previous work also made use of various types of contextual information, such as agreement/disagreement between posts or speeches, author identity, conversation structure in debating forums, and discourse structure. Collective classification has been shown to improve performance (Thomas et al., 2006; Yessenalina et al., 2010; Burfoot et al., 2011; Hasan and Ng, 2013; Walker et al., 2012a; Sridhar et al., 2014).

The setting of ad-hoc claim retrieval, which we address in this work, is different in several respects. First, topics are not known in advance. They may be arbitrarily complex, and belong to any domain. Second, much of the contextual information that was exploited in previous work is not available in this setting. In addition, claims are short sentences, while previous work typically addressed text spanning one or more paragraphs. Moreover, since we may want to present to the user only claims for which we are confident about stance, reliable confidence ranking of our predictions is important. We explore this aspect in our evaluation.

Consequently, our approach relies on generic sentiment analysis, rather than on topic or domain-specific features. We focus on precise semantic analysis of the debate topic and the claim, including target identification, and contrast detection between the claim and the topic targets. While sentiment analysis is a well-studied task, open-domain target identification and open-domain contrast detection between two given phrases have received little attention in previous work.

Consistent/contrastive targets were previously discussed by Somasundaran et al. (2009)², who

²Termed *same/alternative* in their paper.

used them in conjunction with discourse relations to improve the prediction of opinion polarity. However, these targets and relations were not automatically identified, but rather taken from a labeled dataset. Somasundaran and Wiebe (2009) considered debates comparing two products, such as *Windows* and *Mac*. In comparison, topics in our setting are not limited to product names, and the scope of contrast we address is far more general.

Cabrio and Villata (2013) employ textual entailment to detect *support/attack* relations between arguments. However, as illustrated in Table 1, claims typically refer to the pros and cons of the topic target, but do not entail or contradict the topic.

A recent related task is the SemEval 2016 tweets stance classification (Mohammad et al., 2016). In particular, in its weakly supervised subtask (Task B), no labeled training data was provided for the single assessed topic (*Donald Trump*). Beyond the obvious differences in language and content between claims and tweets, the setting of this task is rather different from ours: the topic was known in advance to the participants, and an unlabeled corpus of related tweets was provided. Top performing systems took advantage of this setting, and developed offline rules for automatically labeling the domain corpus. In our setting, the topic is not known in advance, and obtaining a large collection of claims for a given topic does not seem feasible.

3 The Claim Polarity Dataset

The IBM argumentative structure dataset published by Aharoni et al. (2014) contains claims and evidence for 33 controversial topics. In this work we used an updated version of this dataset, which includes 55 topics. Topics were selected at random from the debate motions database at the International Debate Education Association (IDEA) website³. Motions are worded as “This house ...”, in the tradition of British Parliamentary debates. Claims and evidence were manually collected from hundreds of Wikipedia articles. The dataset contains 2,394 claims.

By definition, all claims in the dataset either support or contest the topic, and Aharoni et al. give a few examples for *Pro* and *Con* claims in their paper. However, the dataset itself does not include stance annotations. We enhanced the dataset

³<http://idebate.org/>

with polarity annotations as follows. The polarity of each claim with respect to the motion (*Pro/Con*) was assessed by five annotators, and the final label was determined by the majority annotation.⁴ Table 1 shows examples of motions, claims and their pro/con labeling.

4 Semantic Model for Claim Stance Classification

In this section we propose a model for predicting the stance of a claim c towards a topic sentence t .

We assume that c includes a *claim target* x_c , defined as a phrase about which c makes a positive or a negative assertion. Specifically, it is defined as the most explicit and direct sentiment target in the claim. The *claim sentiment* $s_c \in \{-1, 1\}$ is the sentiment of the claim towards its target, where 1 denotes positive sentiment and -1 denotes negative sentiment. Similarly, we define for a topic t the *topic target* x_t and *topic sentiment* s_t .

We say that the claim target x_c is *consistent* with the topic target x_t if the stance towards x_c implies the *same* stance towards x_t . Similarly, x_c and x_t are *contrastive* if the stance towards x_c implies the *opposite* stance towards x_t . The *contrast relation* between x_c and x_t , denoted $\mathcal{R}(x_c, x_t) \in \{-1, 1\}$ is 1 if x_c and x_t are consistent, and -1 if they are contrastive. Using the above definitions, we define the stance relation between c and t as

$$Stance(c, t) = s_c \times \mathcal{R}(x_c, x_t) \times s_t \quad (1)$$

where $Stance(c, t) \in \{-1, 1\}$, 1 indicates *Pro* and -1 indicates *Con*. Rows 1-8 in Table 1 show examples for x_c , s_c , x_t , s_t and $\mathcal{R}(x_c, x_t)$. It is easy to verify that the model correctly predicts the claim polarity for these examples. For instance, row 3 has x_c =“Unity”, x_t =“Multiculturalism”, $s_c = 1$, $\mathcal{R}(x_c, x_t) = -1$, $s_t = 1$, and the resulting stance is $1 \times (-1) \times 1 = -1$ (*Con*).

Continuous model: The above model produces binary output (+1/-1). In practice, it would be desirable to obtain confidence ranking of the model predictions, which would allow presenting to the user only the top k predictions, or predictions whose confidence is above some threshold. We therefore implemented a continuous variant of the model, where s_c , s_t , $\mathcal{R}(x_c, x_t)$ and the resulting stance score are all real-valued numbers in $[-1, 1]$.

⁴Note that while we considered the original motion phrasing for Pro/Con labeling, the original dataset only contains motion themes as the topics, e.g. *boxing* for “This house would ban boxing”.

#	Debate Topic (Motion)		Claim	
1	This house believes that advertising is harmful. \ominus	\Leftrightarrow	Marketing promotes consumerism and waste. \ominus	Pro
2	This house would ban boxing . \ominus	\Leftrightarrow	Boxing remains the 8th most deadly sport. \ominus	Pro
3	This house would embrace multiculturalism . \oplus	\nLeftrightarrow	Unity is seen as an essential feature of the nation and the nation-state. \oplus	Con
4	This house supports the one-child policy of the republic of China . \oplus	\nLeftrightarrow	Children with many siblings receive fewer resources. \ominus	Pro
5	This house would build hydroelectric dams . \oplus	\Leftrightarrow	As an alternative energy source, a hydroelectric power source is cheaper than both nuclear and wind power. \oplus	Pro
6	This house believes that it is sometimes right for the government to restrict freedom of speech . \ominus	\Leftrightarrow	Human rights can be limited or even pushed aside during times of national emergency. \ominus	Pro
7	This house would abolish the monarchy . \ominus	\Leftrightarrow	Hereditary succession is outdated. \ominus	Pro
8	This house would unleash the free market \oplus	\nLeftrightarrow	Virtually all developed countries today successfully promoted their national industries through protectionism . \oplus	Con
9	This house supports the one-child policy of the republic of China . \oplus		If, for any reason, the single child is unable to care for their older adult relatives, the oldest generations would face a lack of resources and necessities.	Con

Table 1: Sample topic and claim annotations. Targets are marked in bold. \oplus/\ominus denote positive/negative sentiment towards the target, and $\Leftrightarrow/\nLeftrightarrow$ denote consistent/contrastive targets.

For each real-valued prediction, the class is given by its sign, and the confidence is given by its absolute value.

5 Model Assessment via Manual Data Annotation

We assessed the validity and applicability of the proposed model through manual annotation of the IBM dataset.⁵ The labeled data was also used to train and assess sub-components in the model implementation. This section describes the annotation process and the analysis of the annotation results.

Annotation Process: Each of the 55 topics was annotated by one of the authors for its target x_t and sentiment s_t . x_t was used as an input for the claim annotation task. Each claim was labeled independently by five annotators who were given the definitions for claim target x_c , claim sentiment s_c and the contrast relation $\mathcal{R}(x_c, x_t)$ (cf. Section 4). The annotators were first asked to identify x_c and s_c . If successful, they proceeded to determine $\mathcal{R}(x_c, x_t)$.

The final claim labels were derived from the five individual annotations as follows. First, overlapping claim targets were clustered together. If no cluster contained the majority of the annotations

(≥ 3), then the claim was labeled as incompatible with our model. If a majority cluster was found, we discarded annotations where the target was not in this cluster, and selected x_c , s_c and $\mathcal{R}(x_c, x_t)$ based on the majority of the remaining annotations. We required absolute majority agreement (≥ 3) for s_c and $\mathcal{R}(x_c, x_t)$, otherwise the claim was labeled as incompatible with our model.

Rows 1-8 in Table 1 show some examples of annotated claims in our dataset. Row 9 is an example of a claim that was found incompatible with our model.

Data Annotation Results: Majority cluster was found for 98.5% of the claims, and for 92.5% of the claims, the majority of the annotators agreed on the exact boundaries of the target. 94.4% of the claims were found to be compatible with our model. Furthermore, combining the labels for s_c , $\mathcal{R}(x_c, x_t)$ and s_t as in Equation (1) correctly predicted the Pro/Con labels in the dataset (which were collected independently and were not presented to the annotators) for 99.6% of the compatible claims. Given that the pro/con labels are approximately balanced (55.3% are Pro, 44.7% are Con), this result provides a clear and strong evidence for the applicability and validity of the proposed model. This near-perfect correspondence also indicates the high quality of both *Pro/Con* labels and the model-based annotations.

Similar to pro/con labels, claim sentiment is approximately balanced between positive and negative (55% negative vs. 45% positive). Interestingly, 20% of the compatible claims have a con-

⁵The IBM Debating Technologies group in IBM Research has already released several data resources, found here: https://www.research.ibm.com/haifa/dept/vst/mlta_data.shtml. We aim to release the resource presented in this paper as well, as soon as we obtain the required licenses.

trastive relation with the topic target. Since contrastive targets flip polarity, stance classification would fail in these cases, unless these cases are correctly identified and accounted for. This highlights the importance of contrast classification for claim pro/con analysis. We discuss contrast detection in Section 7.

6 Target Extraction and Targeted Sentiment Analysis

Next, we describe an implementation of the stance classification model. This section provides a concise description of target identification and targeted sentiment analysis. The next section presents in more detail our novel contrast detection algorithm. We assume that for the user, directly specifying the topic target x_t and the topic sentiment s_t (e.g., $\langle \textit{boxing}, \textit{Con} \rangle$) is as easy as phrasing the topic as a short sentence (“*This house would ban boxing*”), in terms of supervision effort. Therefore, we focus on finding x_c and s_c , the claim target and sentiment, and assume that x_t and s_t are given.

6.1 Claim Target Identification

Previous work on *targeted/aspect-based* sentiment analysis focused on detecting in user reviews sentiment towards products and their components (Popescu and Etzioni, 2005; Hu and Liu, 2004b), or considered only named entities as targets (Mitchell et al., 2013). Here we address a more general problem of open domain, generic target identification. Table 1 illustrates the diversity and complexity of claim targets.

We set up the problem of claim target identification as a supervised learning problem, using an $L2$ -regularized logistic regression classifier. Target candidates are the noun phrases in the claim, obtained from its syntactic parse⁶. We create one training example from each such candidate phrase x and claim c in our training set. The feature set is summarized in Table 2. Candidate phrases that exactly match the true target or overlap significantly with it are considered positive training examples, while the other candidates are considered negative examples. We measured overlap using the Jaccard similarity coefficient, defined as the ratio between the number of tokens in the intersection and the union of the two phrases, and considered an over-

⁶We used the ESG parser (McCord, 1990; McCord et al., 2012).

Syntactic and Positional: The dependency relation of x in c ; whether x is a direct child of the root in the dependency parse tree for c ; the minimum distance of x from the start or the end of the chunk containing it.

Wikipedia: whether x is a Wikipedia title, (e.g. <i>human rights</i>)

Sentiment: The dependency relation connecting x to any sentiment phrase in the rest of c . The (Hu and Liu, 2004a) sentiment lexicon was used. For example, <i>Hereditary succession</i> is the sentiment target of <i>outdated</i> , indicated by the subject-predicate relation connecting them (Table 1, row 7).
--

Topic relatedness: Semantic similarity between x and the topic target, e.g. <i>Marketing</i> and <i>advertising</i> (Table 1, row 1). We consider morphological similarity, paths in WordNet (Miller, 1995; Fellbaum, 1998), and cosine similarity of word2vec embeddings (Mikolov et al., 2013).
--

Table 2: Features extracted for a target candidate x in a claim c . Examples are taken from Table 1.

lap of 0.6 or higher as significant overlap⁷. The candidate with the highest classifier confidence is predicted to be the target.

6.2 Claim Sentiment Classification

This component determines the sentiment of the claim towards its target. Given our open-domain setting, and the relatively small amount of training data available, we followed the common practice of lexicon-based sentiment analysis (Liu, 2012, pp. 50–53)⁸. Our method is similar to the one described by Ding et al. (2008), and comprises the following steps:

Sentiment matching: Positive and negative terms from the sentiment lexicon of Hu and Liu (2004a) are matched in the claim.

Sentiment shifters application: Sentiment shifters (Polanyi and Zaenen, 2004) reverse the polarity of sentiment words, and may belong to various parts of speech, e.g. “*not successful*+”, “*prevented success*+”, and “*lack of success*+”. We manually composed a small lexicon of about 160 sentiment shifters. The scope was defined as the k tokens following the shifter word.⁹

Sentiment weighting and score computation: Following Ding et al., sentiment term weight decays based on its distance from the claim target. We used a weight of $d^{-0.5}$, where d is the distance in tokens between the sentiment term and the target. Let p and n be the weighted sums of positive

⁷Determined empirically based on the training set.

⁸Our sentiment analyzer was found to outperform the Stanford sentiment analyzer (Socher et al., 2013) on claims.

⁹We experimentally set $k = 8$ based on the training data.

and negative sentiments detected in the claim, respectively. The final sentiment score is then given by $\frac{p-n}{p+n+1}$, following Feldman et al. (2011).

7 Contrast Classification

The most challenging subtask in our model implementation is determining the contrast relation between the topic target x_t , and the claim target x_c . Previous work has focused on word-level contrast and synonym-antonym distinction (Mohammad et al., 2013; Yih et al., 2012; Scheible et al., 2013). The algorithm presented in this section addresses complex phrases, as well as consistent/contrastive semantic relations that go beyond synonyms/antonyms.

7.1 Algorithm

Consider the targets *atheism* and *denying the existence of God*. The relation between these targets is determined based on the contrastive relation between *God* and *atheism*, which is flipped by the negative polarity towards *God*, resulting in a consistent relation between the targets. We call the pair (*God*, *atheism*) the *anchor pair*, defined as the pair of core phrases that establishes the semantic link between the targets.

The following algorithm generalizes this notion, analogously to our claim-level model. The input for the algorithm includes x_c , x_t and a relatedness measure $r(u, v) \in [-1, +1]$ over pairs of phrases u and v . Positive/negative values of r indicate a consistent/contrastive relation, respectively, and the absolute value indicates confidence.

First, anchor candidates are extracted from x_c and x_t , as detailed in the next subsection. The anchor pair is selected based on the association strength of each anchor with the debate topic domain, as well as the strength of the semantic relation between the anchors. Term association with the domain is given by a TF-IDF measure $w(x) = tf(x)/df(x)$, where $tf(x)$ is the frequency of x in articles that were identified as relevant to the topic in the labeled dataset, and $df(x)$ is its overall frequency in Wikipedia. We choose in (x_c, x_t) the anchor pair (a_c, a_t) that maximizes $w(u) \times |r(u, v)| \times w(v)$.

The contrast score is then predicted as $p(x_c, a_c) \times r(a_c, a_t) \times p(x_t, a_t)$, where $p(u, v) \in [-1, +1]$ is the polarity towards v in u . Negative polarity is determined by the presence of words such as *limit*, *ban*, *restrict*, *deny* etc. We manu-

ally developed a small lexicon of stance flipping words, which largely overlaps with our sentiment shifters lexicon. We employ several relatedness measures, described in the next subsection, and the contrast scores obtained for these measures are used as features in the contrast classifier, implemented as a random forest classifier.

The above approach can be extended to find the top-K anchor pairs for complex targets. We use $K = 3$ in our experiments. When considering additional anchor pairs beyond the top-ranked pair (a_c, a_t) , we multiply the above contrast score by $sgn(r(b_c, b_t))$ for each such additional pair (b_c, b_t) . Thus, these pairs may affect the sign of the contrast score but not its magnitude. Anchor pair assignment is computed using the Hungarian Method (Kuhn, 1955).

7.2 Contrast Relations

We initially implemented the following known relatedness measures: (i) morphological similarity, (ii) cosine similarity using word2vec embeddings (Mikolov et al., 2013), (iii) reachability in WordNet via synonym-antonym chains (Harabagiu et al., 2006) and (iv) thesaurus-based synonym-antonym relations using polarity-inducing LSA (Yih et al., 2012). Note that the measures (i) and (ii) above take values only in $[0, 1]$, and thus are indicative of similarity but not of contrast. All these measures suffer from two limitations: (a) They only operate at the token level, while our anchors are often phrases (b) Their coverage on our data is insufficient, in particular for contrastive anchors.

We developed a novel relatedness measure that addresses these limitations, and is used in conjunction with the other measures. Our method is based on co-occurrence of the anchor pair with consistent and contrastive cue-phrases. For example, “*vs*”, “*or*” and “*against*” are contrastive cue phrases, while “*and*”, “*like*” and “*same as*” are consistent cue phrases. We compiled a list of 25 cue phrases.

The anchors are matched in a corpus we composed from the union of two complementary sources, which were found particularly effective for this task:

Query logs: We obtained 2.2 billion queries (450 million distinct queries) from the Blekko® search engine. With over a million distinct queries containing the words *vs*, *vs.*, or *versus*, it is an abundant resource for detecting contrast. Some

examples are: “*God or atheism*”, “*political correctness vs freedom of speech*”, “*free trade vs protectionism*” and “*advertising and marketing*”.

Wikipedia headers: We considered article titles, and section and subsection headers in Wikipedia (3 million in total). For example, “*Military intervention vs diplomatic solution*”.

Compared to full sentences, both queries and headers are short, concise texts, and therefore are less likely to suffer from contextual errors (in which the context alters the meaning of the matched pattern).

The score returned by our method is calculated as follows. Let Lex^+ and Lex^- be the lexicons of consistent and contrastive cue phrases, respectively. Let $Freq(u, v)$ be the number of documents (queries or headers), which contain u and v separated by at most 3 tokens, and $Freq(u, Lex^+, v)$ is the size of the subset of these documents, which also contain a consistent cue phrase between u and v . We then define the probability $P(Lex^+|u, v)$ as $\frac{Freq(u, Lex^+, v)}{Freq(u, v)}$. $P(Lex^-|u, v)$ is defined analogously for the contrastive lexicon. The returned score is $P(Lex^+|u, v)$ if $P(Lex^+|u, v) > P(Lex^-|u, v)$, and $-P(Lex^-|u, v)$ otherwise. We also experimented with other scoring methods, based on pointwise mutual information between the concurrences of the the pair (u, v) and the lexicon cue phrase, as well as statistical significance tests for their co-occurrence. However, the above method was found to perform best on our data.

Generating anchor candidates: Candidate anchors for measures (i)-(iv) are all single tokens. For our method, we additionally considered phrases as anchors. Candidates were generated from diverse sources, including the output of the ESG syntactic parser (McCord, 1990; McCord et al., 2012), the TagMe Wikifier (Ferragina and Scaiella, 2010), named entities recognized with the Stanford NER (Finkel et al., 2005) and multiword expressions in WordNet. Candidates subsumed by larger candidates were discarded. Following Levy et al. (2015), we kept only dominant terms with respect to the topic, by applying a statistical significance test (Hyper-geometric test with Bonferroni correction).

Overall, our method detects many consistent and contrastive pairs missed by previous methods.

7.3 Classification Output

The contrast classifier outputs a score in the $[0, 1]$ interval indicating the likelihood of x_t and x_c being consistent. We found that while it still cannot predict reliably contrastive targets, this consistency confidence score performs well on ranking the targets according to their likelihood of being consistent. We therefore use this score to re-rank our predictions, so that claims that are likely to have consistent targets would rank higher.

8 Evaluation

8.1 Experimental Setup

We evaluated the overall performance of the system, as well as the performance of individual components. The dataset was randomly split into a training set, comprising 25 topics (1,039 claims), and a test set, comprising 30 topics (1,355 claims). The training set was used to train the target identification classifier and the contrast classifier in our system, as well as the baselines described below.

We explore the trade-off between presenting high-accuracy predictions to the user, and making predictions for a large portion of the claims. This tradeoff is controlled by setting a threshold on the prediction confidence, and discarding predictions below that threshold. Let $\#claims$ be the total number of claims. Given some threshold α , we define $\#predicted(\alpha)$ as the number of corresponding predictions, and $\#correct(\alpha)$ as the number of correct predictions. We then define: $coverage(\alpha) = \frac{\#predicted(\alpha)}{\#claims}$, and $accuracy(\alpha) = \frac{\#correct(\alpha)}{\#predicted(\alpha)}$.

We consider the macro averaged $accuracy(\alpha)$ and $coverage(\alpha)$ over the test topics. Our evaluation focuses on the following question: suppose that we require a minimum coverage level, what is the highest accuracy we can obtain? The result is determined by an exhaustive search over threshold values. This assessment was performed for varying coverage levels.

The following configurations were evaluated. The first two configurations represent known strong baselines in stance classification (cf. Section 2).

Unigrams SVM: SVM with unigram features. The SVM classifier gets the claim as an input, and aims to predict the claim sentiment s_c . Assuming consistent targets ($\mathcal{R}(x_c, x_t) = 1$), stance is then predicted as $s_c \times s_t$, where s_t is the given topic

Configuration	Accuracy@Coverage									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Baselines										
Unigrams SVM	0.688	0.688	0.659	0.612	0.587	0.563	0.560	0.554	0.554	0.547
Unigrams+Sentiment SVM	0.717	0.717	0.717	0.709	0.693	0.691	0.687	0.668	0.655	0.632
Our System										
Sentiment Score	0.752	0.720	0.720	0.720	0.720	0.720	0.636	0.636	0.636	0.636
+Targeted Sentiment	0.770	0.770	0.770	0.749	0.734	0.734	0.706	0.632	0.632	0.632
+Contrast Detection	0.849	0.847	0.836	0.793	0.767	0.740	0.704	0.632	0.632	0.632
Our System+Unigrams SVM	0.784	0.758	0.749	0.743	0.730	0.711	0.682	0.671	0.658	0.645

Table 3: Stance classification results. Majority baseline accuracy: 51.9%

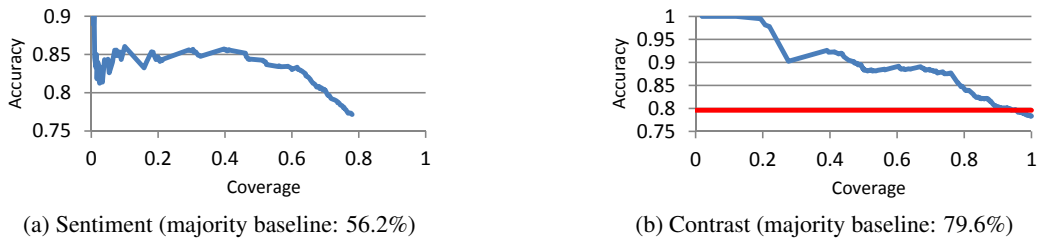


Figure 1: Performance of Sub-Components

sentiment.

Unigrams+Sentiment SVM: The unigram SVM with additional sentiment features. We employed here a simplified version of the sentiment analyzer (cf. Section 6.2), in which target identification is not performed, and sentiment terms are weighted uniformly. The following three features were used: the sums of positive and negative sentiments (p and n), and the final sentiment score.

The next three configurations are incremental implementations of our system. For each configuration, only the difference from the previous configuration is specified.

Sentiment Score: Predicts s_c as the sentiment score of the simplified sentiment analyzer. Stance is predicted as $s_c \times s_t$, similar to the SVM baselines.

+Targeted Sentiment: Employs the targeted sentiment analyzer described in Section 6.2.

+Contrast Detection: Full implementation of our model. Stance score is further multiplied by the output of the contrast classifier, $\mathcal{R}(x_c, x_t)$, predicted for the extracted claim target x_c and the topic target x_t . As discussed in the previous section, this aims to rank higher claims with consistent targets.

Lastly, we tested a combination of our system with the unigrams SVM baseline.

Our System+Unigrams SVM: Adding the targeted sentiment score as a feature to the unigrams SVM. The SVM output is multiplied by the contrast classifier score.

For each configuration, if the classifier outputs zero¹⁰, we predict the majority class in the train set with a constant, very low confidence.

8.2 Results, Analysis and Discussion

The results are shown in Table 3. Comparing the two baselines highlights the importance of sentiment in our open-domain setting, in which no topic-specific training data is available.

Using only the simple sentiment score outperforms the baselines for coverage rates ≤ 0.6 . For higher coverage rates the performance drops from 72% to 63.6%. This happens since the sentiment analyzer makes predictions for 69.4% of the claims, and the remaining claims are given the majority class with a fixed low confidence, as described above. For coverage rates ≥ 0.7 , these claims are added together (since they all match the same threshold), and thus accuracy is actually computed over the whole test set.

Targeted sentiment analysis improves over the non-weighted *Sentiment Score* baseline. It makes predictions for 77.4% of the claims¹¹, and similar to the previous configuration, accuracy drops accordingly from 70.6% to 63.2% for higher coverage rates (≥ 0.8).

Re-ranking based on target consistency confi-

¹⁰This can happen, for example, if the sentiment analyzer does not match any sentiment term in the claim.

¹¹Coverage is improved since sentiment weighting breaks ties between positive and negative sentiments, which result in zero predictions of the simple analyzer.

dence substantially improves accuracy for lower coverage rates (≤ 0.6). For instance, the classifier achieves accuracy of 79.3% over 40% the claims, and 83.6% for 30% of the claims.

Finally, combining our system with the uni-grams SVM allows the classifier to make predictions for claims that are not covered by the targeted sentiment analyzer, and consequently this configuration achieves the best accuracy for high coverage rates (≥ 0.8). It outperforms the SVM baselines for both low and high coverage rates.

Overall, the results confirm that our modular approach outperforms the common practice of monolithic classifiers for stance classification, in particular for making high-accuracy stance predictions for a large portion of the claims. Each component was shown to contribute to the overall performance.

We also assessed the performance for each sub-task on the test set. Claim target identification achieves accuracy of 0.752 for exact matching, and 0.813 for relaxed matching (using the Jaccard measure, as in Section 6.1). Figure 1 shows accuracy vs. coverage curves for targeted claim sentiment analysis and contrast detection. Both components achieve higher accuracy for lower coverage rates, illustrating the effectiveness of their confidence score. As mentioned above, the sentiment analyzer makes a prediction for nearly 80% of the claims, and is shown to perform well. The contrast classifier, while not outperforming the majority baseline over the whole dataset, achieves accuracy that is much higher than the baseline for lower coverage rates.

9 Conclusion

This work is the first to address claim stance classification with respect to a given topic. We proposed a model that breaks down this complex task into simpler, well defined subtasks. Extensive data annotation and analysis has confirmed the applicability and accuracy of this reduction. The annotated dataset, which we plan to share with the community, is another contribution of this work.

The work also presented a concrete implementation of our model, using the collected labeled data to train each component, and demonstrated its effectiveness empirically. We plan to improve each of these components in future work.

Acknowledgments

We would like to thank Yonatan Bilu, Ido Dagan, and Charles Jochim for their helpful feedback on this work.

References

- Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, Baltimore, Maryland, June. Association for Computational Linguistics.
- Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1506–1515, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Elena Cabrio and Serena Villata. 2013. A natural language bipolar argumentation approach to support users in online debate interactions. *Argument & Computation*, 4(3):209–230.
- Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08*, pages 231–240, New York, NY, USA. ACM.
- Adam Faulkner. 2014. Automated classification of stance in student essays: An approach using stance target information and the Wikipedia link-based measure. In *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference, FLAIRS*.
- Ronen Feldman, Benjamin Rosenfeld, Roy Bar-Haim, and Moshe Fresko. 2011. The Stock Sonar - sentiment analysis of stocks based on a hybrid approach. In *Innovative Applications of Artificial Intelligence (IAAI-11)*.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.
- Paolo Ferragina and Ugo Scaiella. 2010. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10*, pages 1625–1628, New York, NY, USA. ACM.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs

- sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06*, pages 755–762. AAAI Press.
- Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356, Nagoya, Japan, October. Asian Federation of Natural Language Processing.
- Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pages 168–177, New York, NY, USA. ACM.
- Minqing Hu and Bing Liu. 2004b. Mining opinion features in customer reviews. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*.
- H. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(3).
- Namhee Kwon, Liang Zhou, Eduard Hovy, and Stuart W. Shulman. 2007. Identifying and classifying subjective claims. In *Proceedings of the 8th Annual International Conference on Digital Government Research: Bridging Disciplines & Domains, dg.o '07*, pages 76–81. Digital Government Society of North America.
- Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.
- Ran Levy, Liat Ein-Dor, Shay Hummel, Ruty Rinott, and Noam Slonim. 2015. Tr9856: A multi-word term relatedness benchmark. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 419–424, Beijing, China, July. Association for Computational Linguistics.
- Marco Lippi and Paolo Torrioni. 2015. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 185–191.
- Bing Liu. 2012. Sentiment analysis and opinion mining. *Morgan & Claypool Publishers*.
- Michael C McCord, J William Murdock, and Branimir K Boguraev. 2012. Deep parsing in watson. *IBM Journal of Research and Development*, 56(3/4):3.1–3.15.
- Michael C. McCord. 1990. Slot grammar: A system for simpler construction of practical natural language grammars. In R. Studer, editor, *Natural Language and Logic: Proc. of the International Scientific Symposium, Hamburg, FRG*, pages 118–145. Springer, Berlin, Heidelberg.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13*, pages 3111–3119, USA. Curran Associates Inc.
- George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.
- Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2013. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.
- Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California, June. Association for Computational Linguistics.
- Livia Polanyi and Annie Zaenen. 2004. Contextual valence shifters. In *Working Notes — Exploring Attitude and Affect in Text: Theories and Applications (AAAI Spring Symposium Series)*.
- Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 339–346, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.
- Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word

- space model. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 489–497, Nagoya, Japan, October. Asian Federation of Natural Language Processing.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 226–234, Suntec, Singapore, August. Association for Computational Linguistics.
- Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124, Los Angeles, CA, June. Association for Computational Linguistics.
- Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 170–179, Singapore, August. Association for Computational Linguistics.
- Dhanya Sridhar, Lise Getoor, and Marilyn Walker. 2014. Collective stance classification of posts in online debate forums. In *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*, pages 109–117, Baltimore, Maryland, June. Association for Computational Linguistics.
- Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335, Sydney, Australia, July. Association for Computational Linguistics.
- Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012a. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596, Montréal, Canada, June. Association for Computational Linguistics.
- Marilyn A. Walker, Pranav Anand, Rob Abbott, Jean E. Fox Tree, Craig Martell, and Joseph King. 2012b. That is your evidence?: Classifying stance in online political debate. *Decis. Support Syst.*, 53(4):719–729, November.
- Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1046–1056, Cambridge, MA, October. Association for Computational Linguistics.
- Wen-tau Yih, Geoffrey Zweig, and John Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1212–1222, Jeju Island, Korea, July. Association for Computational Linguistics.

Exploring the Impact of Pragmatic Phenomena on Irony Detection in Tweets: A Multilingual Corpus Study

Jihen Karoui¹, Farah Benamara¹, Véronique Moriceau²,
Viviana Patti³, Cristina Bosco³, and Nathalie Aussenac-Gilles¹

¹IRIT, CNRS, Université de Toulouse, France

²LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, France

³Dipartimento di Informatica, University of Turin, Italy

¹{karoui, benamara, aussenac}@irit.fr

²{moriceau}@limsi.fr

³{patti, bosco}@di.unito.it

Abstract

This paper provides a linguistic and pragmatic analysis of the phenomenon of irony in order to represent how Twitter's users exploit irony devices within their communication strategies for generating textual contents. We aim to measure the impact of a wide-range of pragmatic phenomena in the interpretation of irony, and to investigate how these phenomena interact with contexts local to the tweet. Informed by linguistic theories, we propose for the first time a multi-layered annotation schema for irony and its application to a corpus of French, English and Italian tweets. We detail each layer, explore their interactions, and discuss our results according to a qualitative and quantitative perspective.

1 Introduction

Irony is a complex linguistic phenomenon widely studied in philosophy and linguistics (Grice et al., 1975; Sperber and Wilson, 1981; Utsumi, 1996). Glossing over differences across approaches, irony can be defined as an incongruity between the literal meaning of an utterance and its intended meaning. For many researchers, irony overlaps with a variety of other figurative devices such as satire, parody, and sarcasm (Clark and Gerrig, 1984; Gibbs, 2000). In this paper, we use irony as an umbrella term that includes sarcasm, although some researchers make a distinction between them, considering that sarcasm tends to be more aggressive (Lee and Katz, 1998; Clift, 1999).

Different categories of irony have been studied in the linguistic literature such as hyperbole, exaggeration, repetition or change of register (see section 3 for a detailed description). These categories were mainly identified in literary texts (books, po-

ems, etc.), and as far as we know, no one explored them in the context of social media. The goal of the paper is thus four folds: (1) analyse if these categories are also valid in social media contents, focusing on tweets which are short messages (140 characters) where the context may not be explicitly represented; (2) examine whether these categories are linguistically marked; (3) test if there is a correlation between the categories and markers; and finally (4) see if different languages have a preference for different categories.

This analysis can be exploited in a purpose of automatic irony detection, which is progressively gaining relevance within sentiment analysis (Maynard and Greenwood, 2014; Ghosh et al., 2015). In particular, it will bring out the most discriminant pragmatic features that need to be taken into account for an accurate irony detection, therefore helping systems improve beyond standard approaches that still heavily rely on features gleaned from the utterance-internal context (Davidov et al., 2010; Gonzalez-Ibanez et al., 2011; Liebrecht et al., 2013; Buschmeier et al., 2014; Hernández Farías et al., 2016).

To this end, informed by well-established linguistic theories of irony, we propose for the first time:

- A multi-layered annotation schema in order to measure the impact of a wide-range of pragmatic phenomena in the interpretation of irony, and to investigate how these phenomena interact with context local to the tweet. The schema includes three layers: (1) *irony activation types* according to a new perspective of how irony activation happens—explicit vs. implicit, (2) *irony categories* as defined in previous linguistic studies, and (3) *irony markers*.
- A multilingual corpus annotated according

to this schema. As the expression of irony is very dependent on culture, we chose, for this first study, three Indo-European languages whose speakers share quite the same cultural background: French, English and Italian. The corpus is freely available for research purposes and can be downloaded here <http://github.com/IronyAndTweets/>.

- A qualitative and quantitative study, focusing in particular on the interactions between irony activation types and markers, irony categories and markers, and the impact of external knowledge on irony detection. Our results demonstrate that implicit activation of irony is a major challenge for future systems.

The paper is organised as follows. We first present our data. Sections 3 and 4 respectively detail the annotation scheme and the annotation procedure. Section 5 discusses the reliability study whereas Section 6 the quantitative results. In Section 7, we compare our scheme to already existing schemes for irony stressing the originality of our approach and the importance of the reported results for automatic irony detection. Finally we end the paper by showing how the annotated corpora are actually exploited in automatic irony detection shared tasks.

2 Data

The datasets used in this study are tweets about hot topics discussed in the media. Our intuition behind choosing such topics is that the pragmatic context needed to infer irony is more likely to be understood by annotators compared to tweets that relate personal content. We relied on three corpora in French, English and Italian, referred to as F , E and I respectively. Table 1 shows the distribution of ironic vs. non ironic tweets in the data.

Corpus	<i>Ironic</i>	<i>Not Ironic</i>
F	2,073	16,179
E	5,173	6,116
I	806 (Sentipolc) + 2,273 (TW-SPINO)	5,642 (Sentipolc)

Table 1: Distribution of tweets in each corpus.

The selection of ironic vs non-ironic tweets has been based on partly different criteria for the three

addressed languages in order to tackle their features.

In English and French, users employ specific hashtags (*#irony*, *#sarcasm*, *#sarcastic*) to mark their intention to be ironic. These hashtags have been often used as gold labels to detect irony in a supervised learning setting. Although this approach cannot be generalized well since not all ironic tweets contain hashtags, it has however shown to be quite reliable as good inter-annotator agreements (kappa around 0.75) between annotators' irony label and the reference irony hashtags have been reported (Karoui et al., 2015). Nevertheless, irony corpus construction through hashtag filtering is not always possible for all languages. For instance, both in Czech and Italian, Twitter users generally do not use the sarcasm (i.e. ‘#sarkasmus’, in Czech; ‘#sarcasmo’ in Italian) or irony (‘#ironie’ in Czech or ‘#ironia’ in Italian) hashtag variants to mark their intention to be ironic, thus in such cases relying on simple self-tagging for collecting ironic samples is not an option (Ptáček et al., 2014; Bosco et al., 2013). Similar considerations hold for Chinese (Tang and Chen, 2014). For what concerns Italian, we observe that even if occasionally Italian tweeters do use creative hashtags to explicitly mark the presence of irony, no generic shared hashtags have been used for long-time which can be considered as firmly established indicators of irony like those used for English.

The corpora built for English and French are new datasets built using the Twitter API as follows. We first selected 9 topics (politics, sport, artists, locations, Arab Spring, environment, racism, health, social media) discussed in the French media from Spring 2014 until Autumn 2015 and in the American media from Spring 2014 until Spring 2016. For each topic, we selected a set of keywords with and without hashtag: politics (e.g. Hollande, Obama), sport (e.g. #Zlatan, #FIFAworldcup), etc. Then, we selected ironic tweets containing the topic keywords and the French (English) ironic hashtags. Finally, we selected non ironic tweets that contained only the topic keywords without the ironic hashtag. We removed duplicates, retweets and tweets containing pictures which would need to be interpreted to understand the ironic content. For English, since we were interested in ironic tweets for our annotation purpose, we stopped collecting messages when the number of ironic tweets was sufficient; this ex-

plains the fact that classes of Ironic and Not Ironic tweets in the English dataset are pretty balanced, i.e. the amount of ironic tweets is not very low compared with the amount of not ironic ones.

Italian data are instead extracted from two existing annotated data: the Sentipolc corpus, released for the shared task on sentiment analysis and irony detection in Twitter at Evalita 2014 (Basile et al., 2014), and TW-SPINO which extends the Spinoza section of the Senti-TUT corpus (Bosco et al., 2013). The Sentipolc dataset is a collection of Italian tweets derived from two existing corpora Senti-TUT and TWITA (Basile and Nissim, 2013). It includes Twitter data exploiting specific keywords and hashtags marking political topics. In Sentipolc, each tweet has an annotation label among five mutually exclusive labels: positive opinion, negative opinion, irony, both positive and negative, and objective. TW-SPINO instead is from the Twitter section of Spinoza¹, a popular collective Italian blog that publishes posts with sharp satire on politics. Since there is a collective agreement about the fact that these posts include irony mostly about politics, they represent a natural way to extend the sampling of ironic expressions. Moreover, while Sentipolc collects tweets spontaneously posted by Italian Twitter users, Spinoza’s posts are selected and revised by an editorial staff, which explicitly characterize the blog as satiric. Such difference will possibly have a reflection on the types and variety of irony we detect in the tweets.

3 A multi-layered annotation schema for irony in social media

To define our annotation schema, we analyzed the different categories of irony studied in the linguistic literature. Several categories have been proposed, as shown in the first column of Table 2. Since all these categories have been found in a specific genre (literary texts), the first step was to check their presence on a small subset of 150 ironic tweets from our corpus. Three observations resulted from this first step, regarding irony activation, irony categories, and irony markers.

3.1 Irony activation

We observed that incongruity in ironic tweets often consists of at least two propositions (or words) P_1 and P_2 which are in contradiction to each other

(i.e. $P_2 = \text{Contradiction}(P_1)$). It is the presence of this contradiction that activates irony. This contradiction can be at a semantic, veracity or intention level. P_1 and P_2 can be both part of the internal context of an utterance (that is explicitly lexicalized), or one is present and the other one implied. We thus defined two types of irony activation: EXPLICIT and IMPLICIT.

In EXPLICIT activation, one needs to rely exclusively on the lexical clues internal to the utterance, like in (1) where there is a contrast between P_1 that contains no opinion word, and P_2 which refers to a situation which is commonly judged as being negative, but in a communicative context which is clearly unsuitable w.r.t. to the one expressed in P_1 .

(1) *L’Italia [attende spiegazioni] $_{P_1}$ da così tanti paesi che comincio a pensare che le nostre richieste [finiscano nello spam] $_{P_2}$.*
(Italy is [waiting for explanations] $_{P_1}$ from so many countries that I suspect our requests are being [labeled as spam] $_{P_2}$.)

Example (2) shows another example of explicit semantic contradiction between P_1 and P_2 .

(2) *Ben non ! [Matraquer et crever des yeux] $_{P_1}$, [ce n’est pas violent et ça respecte les droits] $_{P_2}$!!! #ironie*
(Well, no ! [Clubbing and putting up eyes] $_{P_1}$, [it is not violent and it does respect human rights] $_{P_2}$!!! #irony)

On the other hand, IMPLICIT activation arises from a contradiction between a lexicalized proposition P_1 describing an event or state and a pragmatic context P_2 external to the utterance in which P_1 is false, not likely to happen or contrary to the writer’s intention. The irony occurs because the writer believes that his audience can detect the disparity between P_1 and P_2 on the basis of contextual knowledge or common background shared with the writer. For example, in (3), the negated fact in P_1 helps to recognize that the tweet is ironic.

(3) *La #NSA a mis sur écoute un pays entier. Pas d’inquiétude pour la #Belgique: [ce n’est pas un pays entier.] $_{P_1}$ #ironie*
(The #NSA wiretapped a whole country. No worries for #Belgium: [it is not a whole country.] $_{P_1}$ #irony)
→ P_2 : Belgium is a country.

¹<http://www.spinoza.it/>

State of the art irony categories	Our categories	Usage
<i>Metaphor</i> (Ritchie, 2005; Burgers, 2010)	Analogy ^{Both} : Metaphor and Comparison	Covers analogy, simile, and metaphor. Involves similarity between two things that have different ontological concepts or domains, on which a comparison may be based
<i>Hyperbole</i> (Berntsen and Kennedy, 1996; Mercier-Leca, 2003; Didio, 2007)	Hyperbole/ Exaggeration ^{Both}	Make a strong impression or emphasize a point
<i>Exaggeration</i> (Didio, 2007)		
<i>Euphemism</i> (Muecke, 1978; Seto, 1998)	Euphemism ^{Both}	Reduce the facts of an expression or an idea considered unpleasant in order to soften the reality
<i>Rhetorical question</i> (Barbe, 1995; Berntsen and Kennedy, 1996)	Rhetorical question ^{Both}	Ask a question in order to make a point rather than to elicit an answer (P_1 : asking a question to have an answer, P_2 : no intention to have an answer because it is already known)
<i>Context shift</i> (Haiman, 2001; Leech, 2016)	Context Shift ^{Exp}	A sudden change of the topic/frame, use of exaggerated politeness in a situation where this is inappropriate, etc.
<i>False logic or misunderstanding</i> (Didio, 2007)	False assertion ^{Imp}	A proposition, fact or an assertion fails to make sense against the reality
<i>Oxymoron</i> (Gibbs, 1994; Mercier-Leca, 2003)	Oxymoron/ paradox ^{Exp}	Equivalent to “False assertion” except that the contradiction is explicit
<i>Paradox</i> (Tayot, 1984; Barbe, 1995)		
<i>Situational irony</i> (Shelley, 2001; Niogret, 2004)	Other ^{Both}	Humor or situational irony (irony where the incongruity is not due to the use of words but to a non intentional contradiction between two facts or events)
Surprise effect, repetition, quotation marks, emoticons, exclamation, capital letter, crossed-out text, special signs (Haiman, 2001; Burgers, 2010)	Markers	Words, expressions or symbols used to make a statement ironic

Table 2: Irony categories in our annotation schema.

Note that inferring irony in both types of activation requires some pragmatic knowledge. However, in case of IMPLICIT, the activation of irony happens *only* if the reader knows the context. To help annotators identify irony activation type, we apply the following rule: if P_1 and P_2 can be found in the tweet, then EXPLICIT, otherwise IMPLICIT.

3.2 Irony categories

Both explicit and implicit activation types can be expressed in different ways which we call irony categories. After a thorough inspection of how categories have been defined in linguistic literature, some of them were grouped, like *hyperbole* and *exaggeration*, as we observed that it is very difficult to distinguish them in short messages. We also discarded others, since we considered them as markers rather than irony categories (see the last row in Table 2). We finally retain eight categories, as shown in Table 2: Five are more likely to be found in both types of activation (marked *Both*) while three may occur exclusively in a specific type (marked *Exp* for explicit or *Imp* for implicit).

Categories are not mutually exclusive. Example (5) shows a case of implicit irony activation where the user uses a false assertion P_1 and two rhetorical questions.

(5) @infos140 @mediapart Serge Dassault ?

Corruption ? Non ! Il doit y avoir une erreur. [C'est l'image même de la probité en politique]_{P1} #ironie.

(@infos140 @mediapart Serge Dassault? Corruption? No ! There must be an error. [He is the perfect image of probity in politics]_{P1} #irony) → P₂: Serge Dassault is involved and has been sentenced in many court cases.

3.3 Irony markers

As shown in Table 2, linguistic literature considers other forms of irony categories, such as surprise effect, repetition, etc. Having a computational perspective in mind, we preferred to clearly distinguish between *categories of irony* which are pragmatic devices of irony as defined in the previous section, and *irony markers* which are a set of tokens (words, symbols, propositions) that may activate irony on the basis of the linguistic content of the tweet only. This distinction is also motivated by the fact that markers can either be present in distinct irony categories, not present at all, or present in non ironic tweets as well.

Eighteen markers have been selected for our study. Some of them have shown their effectiveness when used as surface features in irony detection such as punctuation marks, capital letters, reporting speech verbs, emoticons, interjections,

negations, opinion and emotion words (Davidov et al., 2010; Gonzalez-Ibanez et al., 2011; Reyes et al., 2013; Karoui et al., 2015). We investigate in addition novel markers (cf. Table 5): **discourse connectives** as they usually mark oppositions, argumentation chains and consequences; **named entities** and **personal pronouns**, as we assume they can be an indicator of the topic discussed in the tweet (media topic vs. a more personal tweet); **URLs** as they give contextual information that may help the reader to detect irony; and finally **false propositions**. These last four markers might be good features for an automatic detection of implicit irony, for example by detecting that an external context is needed. For example, in (2) markers are negations (*no, not*), punctuation (*!, !!!*), opinion word (*violent*) whereas in (3) markers are named entities (*NSA, Belgium*), negation (*no, not*) and false proposition (*it is not a whole country*).

4 Annotation procedure

For each tweet t , the annotation works as follows²:

- (a) Classify t into *Ironic/Not ironic*. In case annotators do not understand the tweet because of cultural references or lack of background knowledge, t can be classified into the *No decision* class. Note that this third class concerns only French and English corpora since the Italian corpus already has annotations for irony (cf. Section 2).
- (b) If t is ironic, define its activation type: Can P_1 and P_2 be found in the tweet? If *yes* then *explicit*, otherwise *implicit*. Then specify the pragmatic devices used to express irony by selecting one or several categories.
- (c) Identify text spans within the tweet that correspond to a pre-defined list of linguistic markers. Markers are annotated whatever the class of t . This is very important for analyzing the correlation between ironic (vs. non ironic) readings and the presence (vs. absence) of these markers.

Linguistic markers were automatically identified relying on dedicated resources for each language (opinion and emotion lexicons, intensifiers, interjections, syntactic parsers for named entities, etc.).

²The annotation manual is available at: github.com/IronyAndTweets/Scheme

In case of missing markers or erroneous annotations, automatic annotations were manually corrected. Also, to ensure that the annotations were consistent with the instructions given in the manual, common errors are automatically detected: ironic tweets without activation type or irony category, absence of markers, etc. Annotators were asked to correct their errors before continuing to annotate new tweets.

In order to evaluate the stability of the schema regarding language variations, we considered first the French set with a total of 2,000 tweets. Such tweets have been randomly selected from the ones collected as described in Section 2. In order to be sure to have a significant amount of ironic samples, 80% of the total tweets to be manually annotated were selected from the ironic set (i.e. tweets explicitly marked with hashtags like #ironie and #sarcasme)³. Three French native speakers were involved. The annotation of the French corpus followed a three-step procedure where an intermediate analysis of agreement and disagreement between the annotators was carried out. Annotators were first trained on 100 tweets, then were asked to annotate separately 300 tweets (this step allows to compute inter-annotator agreements, cf. next section), to finally annotate 1,700 tweets. In the last step, a revised version of the schema was provided. The adjudicated annotations performed in the second step are part of the corpus.

Then we annotated the English and Italian sets in two steps. First, a training phase (100 tweets each) and then the effective annotation, with respectively 550 and 500 tweets. Four native speakers were involved: two for English and two for Italian. All annotators are skilled in linguistics, researchers and PhD students in computational linguistics.

5 Qualitative results

We report on the reliability of the annotation schema on the French data. Among 300 tweets, annotators agreed on 255 tweets (174 ironic and 63 not ironic), among which 18 have been classified as *No decision*. We get a Cohen's Kappa of 0.69 for *Ironic/Not ironic* classification which is a

³Notice that at this stage such hashtags have been removed, and manual annotation have been applied to 2,000 tweets for all the layers foreseen by our schema. In this way, the reliability of self-tagging has been confirmed, and it was possible to identify the presence of irony also in tweets where it was not explicitly marked by hashtags.

very good score. When compared to gold standard labels, we also obtained a good Kappa measure (0.62), which shows that French irony hashtags are quite reliable. We also noticed that more than 90% of the tweets annotated as *No decision* due to the lack of external context, are in fact ironic according to gold labels. We however decided to keep them for the experiments.

For EXPLICIT vs. IMPLICIT, agreement on activation type knowing the tweet ironic obtained a Kappa of 0.65. It was interesting to note that implicit activation is the majority (76.42%). We observed the same tendency in the other languages too (cf. next section). This is an important result that shows that annotators are able to identify which are the textual spans that activate the incongruity in ironic tweets, whether explicit or implicit, and we expect automatic systems to do as good as humans, at best.

Finally, for irony category identification, since the same ironic tweet can belong to several irony categories, we computed agreements by counting, for each tweet, the number of common categories and then dividing by the total number of annotated categories. We obtained 0.56 which is moderate. This score reflects the complexity of the identification of pragmatic devices. When similar devices are grouped together (mainly hyperbole/exaggeration and euphemism, as they are used to make the intended meaning either stronger or weaker), the score increases to 0.60.

6 Quantitative results

The main aim of our corpus-based study is to verify if the different linguistic theories and definitions made on irony can be applied to social media, especially to tweets, and to study its portability to several languages. Besides standard frequencies, we provide the correlations between irony activation types and markers and between categories and markers in order to bring out features that could be used in a perspective of automatic irony detection. In each corpus, all the frequencies presented here are statistically significant from what would be expected by chance using the χ^2 test ($p < 0.05$).

Table 3 gives the total number of annotated tweets and the activation type for ironic tweets. We observe that most irony activation types in the French and English corpora are implicit with respectively 73.01% and 66.28% while in the Italian corpus, explicit activation is the majority. Notice

that the fact the analysis of the Italian dataset results in a different tendency on this respect can be possibly related to the absence of user-generated ironic hashtags, while user explicitly mark the intention to be ironic (see Section 2).

	Ironic		Non Ironic	No decision	Total
	explicit	implicit			
F	394	1066	380	160	2000
E	144	283	99	24	550
I	260	140	100	–	500

Table 3: Number of tweets in annotated corpora in French (F), English (E) and Italian (I).

Table 4 gives the percentage of tweets belonging to each category of irony split according to explicit vs. implicit activation, when applicable. Higher frequencies are in bold font. We note that *oxymoron/paradox* is the most frequent category for explicit irony in French, English and Italian. Concerning implicit irony, *false assertion* and *other* are the most frequent categories in French and English (*other* is the most frequent one in English because a majority of implicit ironic tweets use situational irony, e.g. *Libertarian Ron Paul condemns Bill Clinton for taking advantage of 20y/o but would not support any law to protect her. #Monica*). In Italian, *false assertion*, *analogy* and *other* are the most frequent categories. As classes are not mutually exclusive, there are 64/38 tweets (resp. in French and English) that belong to more than one category for explicit contradiction. The most frequent combinations are *oxymoron/rhetorical question* and *oxymoron/other* for both English and French; *oxymoron/hyperbole* for French and *oxymoron/analogy* for English. Concerning implicit activation, there are 134/62 tweets (resp. in French and English) that belong to more than one category. The most frequent combinations are *false assertion/other* and *false assertion/hyperbole* for both English and French; and *analogy/other* for English⁴.

Table 5 provides the percentage of tweets containing markers for ironic (explicit or implicit) and non ironic tweets (row in gray). In French, intensifiers, punctuation marks and interjections are more frequent in ironic tweets whereas quotations are more frequent in non ironic tweets. In English, discourse connectors, quotations, comparison words and reporting speech verbs are twice as

⁴For what concerns Italian, at the current stage, only the category considered prevalent for implicit/explicit irony activation was annotated.

	Analogy			Context shift			Euphemism			Hyperbole			Rhetorical question			Oxymoron			False assertion			Other		
	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I
Ex	12	17	21	1	6	19	1	1	5	8	2	9	10	15	10	66	81	28	-	-	-	21	6	7
Im	2	13	26	-	-	-	1	1	4	10	7	5	14	1	12	-	-	-	56	20	34	32	65	19

Table 4: Categories in explicit (*Ex*) or implicit (*Im*) activation in French, English and Italian (in %).

frequent in ironic tweets as in non ironic tweets whereas is it the opposite for personal pronouns. Note that there is no English ironic tweet containing URL since they were all annotated as *no decision* because of a lack of knowledge from the annotators who did not understand the tweet and the Web page pointed by the URL. In Italian, most of markers are more frequent in ironic tweets, while some, like quotations and URL, are more frequent in non ironic tweets⁵.

Our study of negation as an irony marker actually considers negation words like *no* and *not* as well as periphrastic forms of negation such as *ne ... pas* in French. We however excluded lexical negations such as *unreliable*, *unhappy*, *etc.* We will further refine our analysis by considering more words that introduce negation. Also, regarding personal pronouns, they are more common in French and English than in Italian. Italian being a pro-drop language can in part motivate the difference detected with respect to pronouns.

Then, we investigated the correlation between irony markers and irony activation types (resp. between irony markers and irony categories). Our aim is to analyze to what extent these markers can be indicators for irony prediction. Using the Cramer’s V test (Cohen, 1988) on the number of occurrences of each marker, we found a statistically significant ($p < 0.05$) large correlation between markers and ironic/not ironic class for French ($V = 0.156$, $df = 14$) and Italian ($V = 0.31$, $df = 6$); between medium and large for English ($V = 0.132$, $df = 9$). We also found a large correlation between markers and irony activation types for French ($V = 0.196$, $df = 16$), between medium and large for Italian ($V = 0.138$, $df = 5$) and medium for English ($V = 0.083$, $df = 12$).⁶

We also analyzed the correlations per marker ($df=1$). The markers which are the most corre-

⁵For Italian, only values for markers automatically identified reliably, without need of manual correction, are reported (e.g. emoticons, negations). Values for other markers are currently missing since they require a manual check, for instance the case of capital letters, because of the presence in the Italian corpus where all the letters are capital.

⁶For both settings, frequencies < 5 were removed.

lated to ironic/non ironic class are: negations, interjections, named entities and URL for French ($0.140 < V < 0.410$); negations, discourse connectors and personal pronouns for English ($0.120 < V < 0.170$); and quotations, named entities and URL for Italian ($0.310 < V < 0.416$). The markers which are the most correlated to explicit/implicit activation are: opposition markers, comparison words and false assertion for French ($0.140 < V < 0.190$); opposition markers and discourse connectors for English ($0.110 < V < 0.120$); and discourse connectors, punctuation and named entities for Italian ($0.136 < V < 0.213$). Note that even if opinion words are very frequent in ironic tweets, they are however not correlated with either irony/non irony classification or explicit/implicit activation ($V < 0.06$), as many non ironic tweets also contain sentiment words.

Finally, when analyzing which markers are correlated to irony categories, the more discriminant markers are: intensifiers, punctuation, false assertion and opinion words for French (large Cramer’s V); negations, discourse connectors and personal pronouns for English (medium Cramer’s V); and punctuation, interjections and named entities for Italian (medium Cramer’s V).

7 Related work

Most state of the art approaches rely on automatically built social media data collections to detect irony using a variety of features gleaned from the utterance-internal context going from n-gram models, stylistic, to dictionary-based features (Burfoot and Baldwin, 2009; Davidov et al., 2010; Tsur et al., 2010; Gonzalez-Ibanez et al., 2011; Liebrecht et al., 2013; Joshi et al., 2015; Hernández Farías et al., 2015). In addition to the above more lexical features, many authors point out the contribution of pragmatic features, such as the use of common vs. rare words or synonyms (Barbieri and Saggion, 2014). Recent work explores other kinds of contextual information like author profiles, conversational threads, or querying external sources of information (Bamman and Smith, 2015; Wallace et al., 2015; Karoui et al.,

	Emoticon			Negation			Discourse			Humour #*			Intensifier			Punctuation			False prop.*			Surprise			Modality			Quotation		
	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I
Ex	7	2	1	37	58	15	6	41	29	2	14	-	22	9	2	51	30	14	8	0	-	3	0	-	0	2	3	6	21	3
Im	6	4	7	34	61	9	4	29	16	4	15	-	19	12	0	51	28	5	54	18	-	3	3	-	0	2	6	6	21	6
NI	5	10	0	58	75	9	4	13	18	0	0	-	11	9	0	28	30	17	0	0	-	2	0	-	1	6	3	1	10	26
	Opposition			Capital			Pers. pro.*			Interjection			Comparison*			Named E.*			Report verb			Opinion			URL*					
	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I
Ex	9	18	4	3	8	-	31	21	5	14	2	11	8	8	4	97	100	65	1	17	0	48	75	-	33	0	10			
Im	3	11	6	2	6	-	31	24	3	12	0	13	2	12	3	91	97	43	1	14	0	41	74	-	29	0	2			
NI	4	14	4	3	3	-	30	40	1	2	2	12	4	6	1	82	88	98	3	7	1	35	68	-	42	0	44			

Table 5: Markers in ironic (*Exp* or *Imp*) and non ironic (*NI*) tweets in French, English and Italian (in %). Markers with an * have not been studied in irony literature.

	Negation			Discourse			Humour #*			Intensifier			Punctuation			False prop.*			Modality			Quotation				
	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I		
Analogy	46	56	2	6	29	8	6	15	-	21	10	0	49	24	2	13	8	-	0	3	2	0	24	1		
Context sh.	40	100	3	0	11	3	0	0	-	0	0	1	60	44	1	0	0	-	0	11	0	0	44	0		
Euphemism	50	67	1	6	0	2	0	0	-	50	33	0	72	0	1	44	0	-	0	33	0	0	0	1		
Hyperbole	25	42	1	5	25	2	3	8	-	57	38	0	56	21	2	53	46	-	0	0	0	8	4	0		
Rhet. ques.	43	70	2	2	36	3	2	17	-	17	9	0	93	86	1	9	3	-	0	3	0	7	23	1		
Oxymoron	35	59	3	4	43	6	0	14	-	21	10	1	49	26	2	11	0	-	0	2	1	5	20	0		
False asser.	18	57	1	4	25	3	3	7	-	10	16	0	29	14	2	95	89	-	0	0	0	4	16	1		
Other	26	62	2	5	31	3	5	18	-	15	11	0	45	20	2	11	3	-	0	2	1	8	25	1		
	Opposition			Pers. pro.*			Interjection			Comparison*			Named E.*			Report verb			Opinion			URL*				
	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E	I	F	E
Analogy	6	11	2	38	19	2	6	0	3	43	42	3	100	100	17	2	16	0	41	68	-	13	0	1		
Context sh.	0	11	1	40	33	1	20	0	2	20	6	0	80	100	8	0	22	0	60	68	-	0	0	1		
Euphemism	0	0	0	22	0	0	6	0	1	0	0	0	94	100	2	0	33	0	56	67	-	22	0	1		
Hyperbole	2	4	0	29	33	1	18	0	2	0	8	0	88	88	6	3	13	0	84	88	-	21	0	1		
Rhet. ques.	3	15	1	31	27	0	13	2	1	2	5	0	90	97	9	1	17	0	45	73	-	25	0	1		
Oxymoron	12	19	1	32	21	0	15	3	2	2	6	0	99	100	10	1	19	0	55	75	-	11	0	2		
False asser.	3	4	1	31	36	1	13	0	1	2	13	1	90	93	8	1	13	0	45	79	-	25	0	0		
Other	2	11	2	29	22	0	10	0	2	1	10	0	91	98	6	1	16	0	32	74	-	30	0	1		

Table 6: Percentage of tweets in each ironic category containing markers in French, English and Italian.

2015).

Compared to automatic irony detection, little efforts have been done on corpus-based linguistic study of irony. Most of these efforts focus on analyzing the impact of irony in feeling expressions and emotions, by manually annotating tweets at both sentiment polarity and irony levels. E.g. Van Hee et al. (2016) distinguish between ironic, possibly ironic, and non-ironic tweets in English and Dutch. For ironic statements, polarity change that causes irony was annotated to specify whether the change comes from an opposition explicitly marked by a contrast between a positive situation and a negative one, an hyperbole, or an understatement. Stranisci et al. (2016) recently extend the Italian Senti-TUT schema (cf. Section 2) to mark the aspects of the topic being discussed in the tweet, as well as the sentiment expressed towards each aspect. Bosco et al. (2016) propose a second extension with the annotation of French tweets using three labels: positive irony, negative irony, and metaphorical expression.

Current state of the art corpus-based studies are mainly oriented to a sentiment analysis perspective on irony, focusing almost exclusively on cap-

turing tweet’s overall sentiment, explicit polarity change, or syntactic irony patterns. We argue in this paper that irony should instead be an object of study by its own by proposing a more linguistic perspective in order to provide a deeper inspection of what are the inferential mechanisms that activate irony, either explicit or implicit, and the correlations between irony types and irony markers. As far as we know, this is the first study that investigates the portability of a wide-range of pragmatic devices in the interpretation of irony to social media data from a multilingual perspective.

8 Exploiting the annotated corpus for automatic irony detection

The French and Italian parts of the annotated corpus have been respectively exploited as datasets for the first irony detection shared tasks DEFT@TALN2017⁷ and for the SENTIPOLC@Evalita shared task on irony detection⁸ in both 2014 and 2016 editions (Basile et al., 2014; Barbieri et al., 2016). In particular, currently only the first layer of the annotation scheme has been

⁷<https://deft.limsi.fr/2017/>

⁸<http://di.unito.it/sentipolc16>

exploited aiming at detecting if a given tweet is ironic or not. The French task is ongoing. For what concerns Italian, in Sentipolc the irony detection task is one three related but independent sub-tasks focusing on subjectivity, polarity and irony detection, respectively. All tweets of the campaign are, therefore, annotated by a multi-layered annotation scheme including tags for all the three dimensions and available on the Task's website. In 2016 SENTIPOLC has been the most participated EVALITA task with a total of 57 submitted runs from 13 different teams. Not surprisingly, results of the 12 systems evaluated for irony detection seem to suggest that the task appears truly challenging. However, organizers observe that its complexity does not depend (only) on the inner structure of irony, but on unbalanced data distribution in Sentipolc (1 out of 7 examples is ironic in the training set, as they reflect the distribution in a realistic scenario) and on the overall availability of a limited amount of examples (probably not sufficient to generalise over the structure of ironic tweets). The plan is to organize an irony detection dedicated task including a larger and more balanced dataset of ironic tweets in future campaigns. In this perspective, it will be also interesting to investigate if the finer-grained annotation layers for irony proposed here can have a role in the annotation scheme proposed for the new task data.

9 Conclusion and future work

In this paper, we proposed a multi-layered annotation schema for irony in tweets and a multilingual corpus-based study for measuring the impact of pragmatic phenomena in the interpretation of irony. The results show that our schema is reliable for French and that it is portable to English and Italian, observing relatively the same tendencies in terms of irony categories and markers. We observed correlations between markers and ironic/non ironic classes, between markers and irony activation types (explicit or implicit) and between markers and irony categories.

These observations are interesting in a perspective of pragmatically and linguistically informed automatic irony detection, since it brings out the most discriminant features. On this line, we plan to accomplish a validation of the schema based on the definition of an automatic classification model built upon such annotated features. Moreover, an interesting challenge could be to apply the annota-

tion schema to a new language also less culturally close to those addressed in this work.

Finally, another perspective is to investigate how the application of our schema can contribute to shed light on the issue of distinguishing between irony and sarcasm. This issue is challenging, and only recently addressed from computational linguistics. In particular, new data-driven arguments for a possible separation between irony and sarcasm emerged from recent work on Twitter data (Sulis et al., 2016). It could be interesting to see the relation between the finer-grained and pragmatic phenomena related to irony investigated in the present study and the higher-level distinction between irony and sarcasm.

References

- David Bamman and Noah A. Smith. 2015. Contextualized sarcasm detection on Twitter. In *Proceedings of the International Conference on Web and Social Media*, ICWSM 2015, pages 574–577.
- Katharina Barbe. 1995. *Irony in context*, volume 34. John Benjamins Publishing.
- Francesco Barbieri and Horacio Saggion. 2014. Modelling Irony in Twitter: Feature Analysis and Evaluation. In *Proceedings of Language Resources and Evaluation Conference (LREC)*, pages 4258–4264.
- Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the Evalita 2016 SENTiment POLarity Classification Task. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016.*, volume 1749 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Valerio Basile and Malvina Nissim. 2013. Sentiment analysis on italian tweets. In *Proceedings of WASSA 2013*, pages 100–107.
- Valerio Basile, Andrea Bolioli, Malvina Nissim, Viviana Patti, and Paolo Rosso. 2014. Overview of the Evalita 2014 SENTiment POLarity Classification Task. In *Proc. of EVALITA 2014*, pages 50–57, Pisa, Italy. Pisa University Press.
- Dorthe Berntsen and John M. Kennedy. 1996. Unresolved contradictions specifying attitudes in metaphor, irony, understatement and tautology. *Poetics*, 24(1):13–29.
- Cristina Bosco, Viviana Patti, and Andrea Bolioli. 2013. Developing Corpora for Sentiment Analysis: The Case of Irony and Senti-TUT. *IEEE Intelligent Systems*, 28(2):55–63, March.

- Cristina Bosco, Mirko Lai, Viviana Patti, and Daniela Virone. 2016. Tweeting and Being Ironic in the Debate about a Political Reform: the French Annotated Corpus TWitter-MariagePourTous. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Clint Burfoot and Timothy Baldwin. 2009. Automatic satire detection: Are you having a laugh? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 161–164, Suntec, Singapore, August. Association for Computational Linguistics.
- Christian Burgers. 2010. *Verbal irony: Use and effects in written discourse*. Ph.D. thesis, Radboud Universiteit Nijmegen.
- Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. 2014. An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49, Baltimore, Maryland, June. ACL.
- Herbert H. Clark and Richard J. Gerrig. 1984. On the pretense theory of irony. *Journal of Experimental Psychology: General*, 113(1):121–126.
- Rebecca Clift. 1999. Irony in conversation. *Language in Society*, 28:523–553.
- Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences Second Edition*. Lawrence Erlbaum Associates.
- Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-Supervised Recognition of Sarcasm in Twitter and Amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116, Uppsala, Sweden, July. Association for Computational Linguistics.
- Lucie Didio. 2007. *Une approche sémiotico-sémiotique de l'ironie*. Ph.D. thesis, Université de Limoges.
- Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment Analysis of Figurative Language in Twitter. In *Proceedings of SemEval 2015, Co-located with NAACL*, page 470478. ACL.
- Raymond W. Gibbs. 1994. *The poetics of mind: Figurative thought, language, and understanding*. Cambridge University Press.
- Raymond W. Gibbs. 2000. Irony in talk among friends. *Metaphor and symbol*, 15(1-2):5–27.
- Roberto Gonzalez-Ibanez, Smaranda Muresan, and Nina Wacholde. 2011. Identifying sarcasm in Twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.
- Herbert Paul Grice, Peter Cole, and Jerry L. Morgan. 1975. Syntax and semantics. *Logic and conversation*, 3:41–58.
- John Haiman. 2001. *Talk is cheap: Sarcasm, alienation, and the evolution of language*. Oxford University Press, USA.
- Delia Irazú Hernández Farías, Emilio Sulis, Viviana Patti, Giancarlo Ruffo, and Cristina Bosco. 2015. Valento: Sentiment analysis of figurative language tweets with irony and sarcasm. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 694–698, Denver, Colorado, June. ACL.
- Delia Irazú Hernández Farías, Viviana Patti, and Paolo Rosso. 2016. Irony Detection in Twitter: The Role of Affective Content. *ACM Transactions on Internet Technologies*, 16(3):19:1–19:24.
- Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 757–762, Beijing, China, July. ACL.
- Jihen Karoui, Farah Benamara, Véronique Moriceau, Nathalie Aussenac-Gilles, and Lamia Hadrich-Belguith. 2015. Towards a contextual pragmatic model to detect irony in tweets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 644–650, Beijing, China, July. ACL.
- Christopher J. Lee and Albert N. Katz. 1998. The differential role of ridicule in sarcasm and irony. *Metaphor and Symbol*, 13(1):1–15.
- Geoffrey N. Leech. 2016. *Principles of pragmatics*. Routledge.
- Christine Liebrecht, Florian Kunneman, and Antal Van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 29–37, Atlanta, Georgia, June. Association for Computational Linguistics.
- Diana Maynard and Mark Greenwood. 2014. Who cares about Sarcastic Tweets? Investigating the Impact of Sarcasm on Sentiment Analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4238–4243, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

- Florence Mercier-Leca. 2003. *L'ironie*. Hachette supérieur.
- Douglas C. Muecke. 1978. Irony markers. *Poetics*, 7(4):363–375.
- Philippe Niogret. 2004. *Les figures de l'ironie dans A la recherche du temps perdu de Marcel Proust*. Editions L'Harmattan.
- Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm Detection on Czech and English Twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 213–223, Dublin, Ireland, August. Dublin City University and ACL.
- Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation*, 47(1):239–268.
- David Ritchie. 2005. Frame-shifting in humor and irony. *Metaphor and Symbol*, 20(4):275–294.
- Ken-ichi Seto. 1998. On non-echoic irony. *Relevance Theory: Applications and Implications*, 37:239.
- Cameron Shelley. 2001. The bicoherence theory of situational irony. *Cognitive Science*, 25(5):775–818.
- Dan Sperber and Deirdre Wilson. 1981. Irony and the use-mention distinction. *Radical pragmatics*, 49:295–318.
- Marco Stranisci, Cristina Bosco, D.I. Hernández Fariás, and Viviana Patti. 2016. Annotating sentiment and irony in the online italian political debate on #labuonascuola. In *Proceedings of LREC 2016*, pages 2892–2899. ELRA.
- Emilio Sulis, D. Irazú Hernández Fariás, Paolo Rosso, Viviana Patti, and Giancarlo Ruffo. 2016. Figurative messages and affect in Twitter: Differences between #irony, #sarcasm and #not. *Knowledge-Based Systems*, 108:132 – 143. New Avenues in Knowledge Bases for Natural Language Processing.
- Yijie Tang and HsinHsi Chen. 2014. Chinese irony corpus construction and ironic structure analysis. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1269–1278.
- Claudine Tayot. 1984. *L'ironie*. Ph.D. thesis, Claude Bernard University (Lyon).
- Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*.
- Akira Utsumi. 1996. A unified theory of irony and its computational formalization. In *Proceedings of COLING, the 16th conference on Computational Linguistics-Volume 2*, pages 962–967. Association for Computational Linguistics.
- Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016. Exploring the Realization of Irony in Twitter Data. In *Proceedings of LREC*. European Language Resources Association (ELRA).
- Byron C. Wallace, Do Kook Choe, and Eugene Charniak. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL-IJCNLP 2015*, pages 1035–1044. ACL.

A Multi-View Sentiment Corpus

Debora Nozza Elisabetta Fersini Enza Messina

University of Milano-Bicocca,

Viale Sarca 336, 20126, Milan, Italy

{debora.nozza, fersini, messina}@disco.unimib.it

Abstract

Sentiment Analysis is a broad task that involves the analysis of various aspect of the natural language text. However, most of the approaches in the state of the art usually investigate independently each aspect, i.e. Subjectivity Classification, Sentiment Polarity Classification, Emotion Recognition, Irony Detection. In this paper we present a Multi-View Sentiment Corpus (MVSC), which comprises 3000 English microblog posts related the *movie domain*. Three independent annotators manually labelled MVSC, following a broad annotation schema about different aspects that can be grasped from natural language text coming from social networks. The contribution is therefore a corpus that comprises five different views for each message, i.e. *subjective/objective*, *sentiment polarity*, *implicit/explicit*, *irony*, *emotion*. In order to allow a more detailed investigation on the human labelling behaviour, we provide the annotations of each human annotator involved.

1 Introduction

The exploitation of user-generated content on the Web, and in particular on the social media platforms, has brought to a huge interest on Opinion Mining and Sentiment Analysis. Both Natural Language Processing (NLP) communities and corporations are continuously investigating on more accurate automatic approaches that can manage large quantity of noisy natural language texts, in order to extract opinions and emotions towards a topic. The data are usually collected from Twitter, the most popular microblogging platform. In this particular environment, the posts, called tweets,

are constrained to a maximum number of characters. This constraint, in addition to the social media context, leads to a specific language rich of synthetic expressions that allow the users to express their ideas or what happens to them in a short but intense way.

However, the application of automatic sentiment classification approaches, in particular when dealing with noisy texts, is subjected to the presence of sufficiently manually annotated dataset to perform the training. The majority of the corpora available in the literature are focused on only one (or at most two) aspects related to Sentiment Analysis, i.e. Subjectivity, Polarity, Emotion, Irony.

In this paper we propose a Multi-View Sentiment Corpus¹, manually labelled by three independent annotators, that makes possible to study Sentiment Analysis by considering several aspects of the natural language text: *subjective/objective*, *sentiment polarity*, *implicit/explicit*, *irony* and *emotion*.

2 State of the Art

The work of Go et al. (2009) was the first attempt to address the creation of a sentiment corpus in a microblog environment. Their approach, introduced in (Read, 2005), consisted to filter all the posts containing emoticons and subsequently label each post with the polarity class provided by them. For example, :) in a tweet indicates that the tweet contains positive sentiment and :(indicates that the tweet contains negative sentiment. The same procedure was also applied in (Pak and Paroubek, 2010), differently from the aforementioned works they introduced the class of objective posts, retrieved from Twitter accounts of popular newspapers and magazines. Davidov et

¹The proposed corpus is available at www.mind.disco.unimib.it

al. (2010) maintained the idea of distant supervision by combining 15 common emoticons and 50 sentiment-driven hashtags for automatic labelling. However, an intervention of human experts was needed to annotate the sentiment of frequent tags. Kouloumpis et al. (2011) extended their work in order to perform a 3-way polarity classification (positive, negative and neutral) on the Edinburgh Twitter corpus (Petrović et al., 2010).

Mohammad (2012) and Wang et al. (2012) applied the same distant supervision approach for the construction of a large corpus for emotion classification. They collect the data retrieving tweets by considering as keywords a predefined list of emotion hashtags. In (Mohammad, 2012), the authors used the Ekman’s six basic emotions (#anger, #disgust, #fear, #joy, #sadness, and #surprise), while in (Wang et al., 2012) the authors expanded this list by including both basic and secondary emotions and their lexical variants, for a total of 131 keywords.

Hashtags have also been used to create datasets for irony detection purposes. The work of Reyes et al. (2013) proposed a corpus of 40000 tweets, 10000 ironic and 30000 non ironic tweets respectively retrieved with the hashtags #irony for the former and #education, #humor, #politics for the latter.

However, each of these resources have been created either fully automatically or in a semi-supervised way based on the assumption that single words and symbols are representative of the whole document. Moreover, the use of hashtags and emoticons for exploiting distant-supervision approaches can definitely create a bias towards posts that do not use these forms of expression to communicate opinions and emotions. Adopting a manual annotation approach is crucial for dealing with these issues and obtaining high quality labelling. In this direction the SemEval corpora (Nakov et al., 2013; Rosenthal et al., 2014; Nakov et al., 2016) have provided a fundamental contribution. These datasets have been labelled by taking advantage of crowdsourcing platforms, such as Amazon Mechanical Turk and CrowdFlower. Although the size of these corpora is very high (around 15-20K posts), Mozetič et al. (2016) overly exceeded these dimensions proposing a set of over 1.6 million sentiment labelled tweets. This corpus, that is the largest manually-labelled dataset reported in the literature, was an-

notated in 13 European languages.

Regarding emotion classification, Roberts et al. (2012) introduced a corpus of tweets manually labelled with the Ekman’s six basic emotions and love. In (Liew et al., 2016), the authors extended their work by considering a fine-grained set of emotion categories to better capture the richness of expressed emotions.

The only manually-annotated corpus on irony detection was proposed by (Gianti et al., 2012). They studied the use of this particular device on Italian tweets, focusing on the political domain.

In this paper, we present a Multi-View Sentiment Corpus (MSVC) on English microblog posts that differs from the state of the art corpora for several reasons:

- The proposed corpus is the first benchmark that collects implicit or explicit opinions. This contribution will allow researchers to develop sentiment analysis approaches able to model opinions not directly expressed.
- The corpus provides different annotations simultaneously: subjectivity/objectivity, polarity, implicitness/explicitness, emotion, irony. This characteristic allows researchers to perform wide-ranging studies on the users’ opinions, instead of considering each of this view as independent from the others.
- The corpus will show the label provided by each annotator, instead of producing a final label obtained by a majority voting rule. Given the different expertise of the annotators involved, a detailed investigation on single behaviours can be performed to improve the knowledge about the annotation procedures.
- This is the first corpus that explicitly labels emojis. We aim to prove that the role of the emojis is strictly related to the context where they appear: their contribution in terms of conveyed sentiment (or conveyed topic) strictly depends on the domain where they are used.

3 Annotation Procedure

The corpus has been annotated by considering different views related to the same post: subjectivity/objectivity, polarity, implicitness/explicitness, presence of irony and emotion. In this section, we provide a definition and examples for each of these

views. Moreover, we present the characteristics of the annotators in order to have more insights on their behaviour.

3.1 Annotation of Subjectivity/Objectivity

Given a post p about a given topic t , its subjectivity or objectivity can be defined as follows (Liu, 2012):

Definition 1. An *objective* post p_o presents some factual information about the world, while a *subjective* post p_s expresses some personal feelings, views, or beliefs.

In microblogs contexts the recognition of objective posts can be easily misled by the presence of hashtags and other linguistic artefacts that aim to show the post as more appealing. The reported examples are very similar, despite they belong to different classes:

[Objective] “Tonight @CinemaX #SuicideSquad!! Come to see #HarleyQuinn :)”

[Subjective] “-1 to #Deadpool...that’s tomorrow!!!! I can’t wait!”

3.2 Annotation of polarity

Given a subjective post p_s that expresses an opinion about a topic t , we want to determine its polarity between positive, negative and neutral classes. While the definition of positive and negative classes is commonly clear, the neutral label is differently treated in the state of the art. As in Pang and Lee (2008), we use neutral only in the sense of a sentiment that lies between positive and negative.

Posts that express a sentiment about specific aspects of a given topic t , such as actors, scenes, commercials for a film are considered part of the topic. Moreover, it is important to understand what is the target of the opinion, because it can lead to completely different interpretations.

[Positive] “Best Joker EVER!! #suicidesquad”

[Negative] “Deadpool is so childish! I slept during the movie”

[Neutral] “Good movie, @VancityReynolds worst actor ever #deadpool” (neutral - mixed sentiment)

[Neutral] “I love my boyfriend! We are watching deadpool tonight” (positive about the boyfriend - neutral about the film)

3.3 Annotation of explicit/implicit opinion

Given a subjective post p_s that expresses an opinion about a topic t , we can define its implicitness or explicitness as follows (Liu, 2012):

Definition 2. An *explicit opinion* is a subjective statement that gives an opinion.

Definition 3. An *implicit (or implied) opinion* is an objective statement that implies an opinion. Such an objective statement usually expresses a desirable or undesirable fact.

The detection of an implicit opinion can be complex because it does not rely on specific words (e.g. amazing, awful), as in the following examples:

[Explicit - Positive] “Suicide Squad is a great movie and an awesome cast”

[Implicit - Positive] “I’ve already watched Deadpool three times this month”

[Implicit - Negative] “I went out the cinema after 15 minutes #suicidesquad”

3.4 Annotation of Irony

Given a subjective post p_s that expresses an opinion about a topic t , the presence of irony can be detected focusing on the definition given by Wilson and Sperber (2007):

Definition 4. *Irony* is basically a communicative act that expresses the opposite of what is literally said.

Irony is one of the most difficult figurative language to comprehend, and a person can perceive it differently depending on several factors (e.g. culture, language).

[Ironic] “Hey @20thcenturyfox remember when you didn’t want anything to do with #Deadpool and now it’s your biggest opening weekend ever?”

3.5 Annotation of Emotion

A post p about the topic t can be associated to an emotion e corresponding to the eight Plutchik primary emotions (shown in Figure 1): anger, anticipation, joy, trust, fear, surprise, sadness and disgust. We provide an example for each emotion.

[Anger] “#Deadpool I wasted time and money grrrrrrrr”

[Anticipation] “Can’t wait to see Deadpool”

[Joy] “Deadpool was A-M- A-Z- I-N- G”

[Trust] “Best movie ever #Deadpool! Trust me!”

[Fear] “Saw #Deadpool last night. I was frightened during some crude scenes!”

[Surprise] “Much to my surprise, I actually liked *Deadpool*.”

[Sadness] “i finally got to watch *deadpool* and im so sad this is so boring”

[Disgust] “*Deadpool* is everything I hate about our century combined in the trashiest movie possible.”

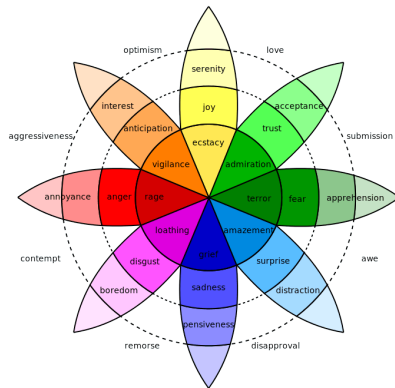


Figure 1: Plutchik’s wheel of emotions.

3.6 Annotation of Emojis

Given a post p related to a specific topic t , each emoji (if present) has been labelled as *positive*, *negative*, *neutral* or *topic-related* according to the context where it has been used. We provide an example for each label.

[Positive] “Love Jared Leto 🤩❤️”

[Negative] “*SuicideSquad* was awful 🤢”

[Neutral] “#*SuicideSquad*...Still thinking 🤔”

[Topic] “*Deadpool* today !! 🍷👤🎥”

3.7 Annotators

The complete set of posts has been labelled by three different annotators. Each annotator is a very proficient English speaker and he/she has a different level of NLP background and topic knowledge from the others. We distinguish these two types of knowledge because they are equally important and necessary for annotating a dataset, especially in a movie domain. A topic expert can be very confident on understanding the meaning of the text, but without any NLP knowledge he/she would not be able to perform a confident annotation, especially when dealing with the implicitness/explicitness and subjectivity/objectivity labels. On the other hand, being only a NLP expert

is not sufficient when in the text subtle and sophisticated references to the topic are present, resulting in an incorrect annotation because of an improper understanding.

The first annotator A_1 is a NLP expert while he/she is not very confident on the topic selected, the second annotator A_2 has a good expertise in NLP and a good knowledge about the topic, the third annotator A_3 is a beginner in the field of NLP but he/she is competent on the topic.

4 Dataset

The data has been retrieved by monitoring different keywords on the Twitter microblogging platform related to two popular movies: *Deadpool* and *Suicide Squad*. This choice was motivated by the intention to increase the number of opinionated posts and therefore to have a variety of aspects to be analysed. Also, both the movies were massive blockbuster successes with popular actors and this led to a very wide and diverse audience.

This case study is experimentally convenient for our purposes because it represents a domain where people are more willing to express opinions, so that the final corpus will have a variety of opinionated tweets expressed in diverse ways. The collection of the data has been performed in the narrow days of the release date, *Deadpool* 18th February 2016 and *Suicide Squad* 1th August 2016.

After the streaming collection phase, we filter out the non-English tweets, duplicates and retweets resulting in a dataset of millions of posts. Then, we randomly sampled 3000 tweets equally distributed between the topics, maintaining the original daily distribution. This sample has been manually annotated, obtaining a final corpus composed of 1500 posts about *Deadpool* and 1500 posts about *Suicide Squad*.

On average, a tweet is composed of about 14 words of which one is a hashtag. Although this number can lead to conclude that hashtags are an important language expression and therefore they can be used for automatically collecting opinions and emotions, we found that most of them are strongly related only to the topic, e.g. #ShowTimeAtGC, #Joker, #HarleyQuinn, #DC. A preliminary analysis of the user mentions has shown that users are inclined to directly mention the actors or the entertainment companies for complaining or complimenting, and this, together with hashtags, can be particularly helpful when per-

forming aspect-based sentiment analysis.

Regarding emojis, the most frequent ones are (as expected) sentiment-driven, i.e. joy, heart eyes, sob.

5 Annotation Evaluations

The annotation of emotions, sentiment and other emotional aspects in a microblog message is not an easy task, and strongly depends on subjective judgement of human annotators. Annotators can disagree between themselves, and sometimes an individual cannot be consistent with her/himself. The disagreement depends on the complexity of the annotation task, the use of complex language (e.g., slang), or simply on the poor annotator work.

Table 1: Label distribution per annotator

	A ₁	A ₂	A ₃
Subjective	0.671	0.748	0.657
Objective	0.330	0.252	0.343
None	0.330	0.252	0.343
Positive	0.509	0.476	0.426
Neutral	0.038	0.146	0.131
Negative	0.123	0.126	0.100
None	0.330	0.252	0.343
Explicit	0.254	0.512	0.416
Implicit	0.416	0.236	0.242
Not Ironic	0.980	0.988	0.971
Ironic	0.020	0.012	0.029
None	0.374	0.355	0.507
Joy	0.317	0.328	0.243
Anticipation	0.144	0.108	0.078
Disgust	0.070	0.071	0.047
Surprise	0.038	0.024	0.038
Sadness	0.035	0.044	0.036
Anger	0.014	0.034	0.028
Trust	0.008	0.034	0.022
Fear	0.001	0.001	0.002

In Table 1, we report some statistics that summarize behaviours of the involved annotators. By analysing the distributions, we can observe different attitudes: A₁ is inclined to label more posts as positive against the neutral ones; A₂ shows a predisposition to identify a high number of explicit expressions; A₃ has a low sensibility to capture the emotions behind the text. Moreover, we can highlight a balanced distribution for implicit/explicit opinions.

For those tweets encoding one of the eight emotions, there is a predominance of the joy label. Concerning the remaining classes the distributions are skewed towards a specific label, i.e. Subjective, Positive and Not Ironic.

An analogous consideration can be drawn for the emoji distribution (see Table 2). It turns out that most of the emojis are positive, especially the most popular ones and their presence provide an insight of the human emotional perceptions.

Table 2: Emojii distribution

	A ₁	A ₂	A ₃
Topic	0.141	0.129	0.095
Positive	0.559	0.601	0.593
Negative	0.141	0.187	0.173
Neutral	0.160	0.083	0.139

By a detailed analysis of the emoji annotations, it emerges that the role of the emojis is closely related to the context where they appear: their contribution in terms of conveyed sentiment (or conveyed topic) strictly depends on the domain where they are used. In Table 3, we report a comparison between the label distribution of two emojis in our corpus and the corresponding distribution in a state of the art emoji sentiment lexicon (Novak et al., 2015).

In the proposed corpus, the *fire* emoji has been mainly labelled as positive because it represents the word “hot”, whose meaning is intended as something beautiful and trendy. However, in the emoji sentiment lexicon the same emoji primarily corresponds to a neutral sentiment. Similar considerations can be drawn for the *pistol* emoji: in our corpus it represents the topic underlying the two movies, while in the state of the art lexicon it is frequently used to denote a negative sentiment orientation. As conclusion, any emoji should be not considered as independent on the context and therefore evaluated according to its semantic.

5.1 Agreement Measures

The kappa coefficient (Cohen, 1960) is the most used statistic for measuring the degree of reliability between annotators. The need for consistency among annotators immediately arises due to the

Table 3: Emoji label distribution



			
Multi-View Sentiment Corpus	Topic	0.127	0.476
	Negative	0.079	0.143
	Neutral	0.111	0.190
	Positive	0.683	0.190
Emoji Sentiment Ranking (Novak et al., 2015)	Negative	0.124	0.493
	Neutral	0.613	0.209
	Positive	0.263	0.298

Table 4: Inter-agreement (PABAK-OS)

	Subjective/Objective	Sentiment Polarity	Implicit/Explicit	Irony	Emotion
\mathbf{A}_1 vs \mathbf{A}_2	0.606	0.598	0.354	0.949	0.590
\mathbf{A}_2 vs \mathbf{A}_3	0.670	0.596	0.476	0.923	0.601
\mathbf{A}_1 vs \mathbf{A}_3	0.592	0.585	0.416	0.912	0.551

Table 5: Self-agreement (PABAK-OS)

	Subjective/Objective	Sentiment Polarity	Implicit/Explicit	Irony	Emotion
\mathbf{A}_1	0.920	0.920	0.640	1.000	0.820
\mathbf{A}_2	0.878	0.867	0.670	0.960	0.865
\mathbf{A}_3	1.000	0.920	0.850	0.878	0.820

variability among human perceptions. This **inter-agreement** measure can be summarized as:

$$k = \frac{\text{observed agreement} - \text{chance agreement}}{1 - \text{chance agreement}} \quad (1)$$

However, considering only this statistic is not appropriate when the prevalence of a given response is very high or very low in a specific class. In this case, the value of kappa may indicate a low level of reliability even with a high observed proportion of agreement. In order to address these imbalances caused by differences in prevalence and bias, Byrt et al. (1993) introduced a different version of the kappa coefficient called prevalence-adjusted bias-adjusted kappa (PABAK). The estimation of PABAK depends solely on the observed proportion of agreement between annotators:

$$PABAK = 2 \cdot \text{observed agreement} - 1 \quad (2)$$

A more reliable measure for estimating the agreement among annotators is PABAK-OS (Parker et al., 2011), which controls for chance agreement. PABAK-OS aims to avoid the peculiar, unintuitive results sometimes obtained from Cohen’s Kappa, especially related to skewed annotations (prevalence of a given label).

We report in Table 4, the inter-agreement between couples of annotators distinguished for each label. We can easily note that the highest agreement is related to the irony/not-irony labelling. This is due to the predominance of non-ironic messages identified by all the annotators. Thus, we perform a detailed analysis on the disagreement between each couple of annotators regarding only the ironic messages. From the results, reported in Table 6, we can confirm that \mathbf{A}_1 and \mathbf{A}_2 annotators are more willing to interpret irony similarly (as already stated in Table 4).

Concerning the implicit/explicit labels, the inter-agreement measure highlights the difficulties

Table 6: Count of disagreement on the irony label

	Disagreement (<i>irony</i>)
\mathbf{A}_1 vs \mathbf{A}_2	76
\mathbf{A}_2 vs \mathbf{A}_3	114
\mathbf{A}_1 vs \mathbf{A}_3	132

encountered by the annotators to distinguish “objective statements” (see Definition 1) from “objective statements that imply an opinion” (see Definition 3). Regarding the remaining labels, we can assert that there is a moderate agreement between the labellers. An analogous conclusion can be derived for the consensus about the emoji annotation, where the inter-agreement is 0.731 for \mathbf{A}_1 vs \mathbf{A}_2 , 0.771 for \mathbf{A}_2 vs \mathbf{A}_3 , and 0.647 for \mathbf{A}_1 vs \mathbf{A}_3 .

When dealing with complex annotations, the perception of the same annotator on the same post can change over time, resulting in inconsistent labelling. In order to estimate the uncertainty of the annotation of each labeller, we sampled a portion of tweets to be annotated twice by the same annotator. We report in Table 5 the **self-agreement** measure, that is a valid index to quantify the quality of the labelling procedure. The resulting statistics show that there is a high self-agreement for almost all the labels. The annotators can be considered moderately reliable for implicit/explicit annotations and very accurate for the remaining labels.

6 Conclusion

In this paper we presented a Multi-View Sentiment Corpus (MVSC), which simultaneously considers different aspects related to sentiment analysis, i.e. *subjectivity*, *polarity*, *implicitness*, *irony*, *emotion*. We described the construction of the corpus, together with annotation schema, statistics and some interesting remarks. The proposed corpus is aimed at providing a benchmark to develop sentiment analysis approaches able to model opinions not directly expressed. Researchers can also take advan-

tage of the complete label set given by the annotators to investigate their behaviours and the underlying annotation procedures. We finally provided some interesting conclusions related to the use of emojis, highlighting that their role is strictly related to the context where they appear. As future work, we aim at defining novel machine learning models able to simultaneously take advantage of the multiple views available. Moreover, an annotation scheme at a fine-grained level will be investigated.

References

- Ted Byrt, Janet Bishop, and John B. Carlin. 1993. Bias, prevalence and kappa. *Journal of Clinical Epidemiology*, 46(5):423–429.
- Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.
- Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics.
- Andrea Gianti, Cristina Bosco, Viviana Patti, Andrea Bolioli, and Luigi Di Caro. 2012. Annotating irony in a novel italian corpus for sentiment analysis. In *Proceedings of the 4th Workshop on Corpora for Research on Emotion Sentiment and Social Signals*, pages 1–7.
- Alec Go, Lei Huang, and Richa Bhayani. 2009. Twitter sentiment analysis. *Final Projects from CS224N for Spring 2008/2009 at The Stanford Natural Language Processing Group*.
- Efthymios Kouloumpis, Theresa Wilson, and Johanna D. Moore. 2011. Twitter sentiment analysis: The good the bad and the omg!. *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, 11:538–541.
- Jasy Suet Yan Liew, Howard R. Turtle, and Elizabeth D. Liddy. 2016. Emotweet-28: A fine-grained emotion corpus for sentiment analysis. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA).
- Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.
- Saif M. Mohammad. 2012. #emotional tweets. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics.
- Igor Mozeti, Miha Grar, and Jasmina Smailovi. 2016. Multilingual twitter sentiment classification: The role of human annotators. *PLOS ONE*, 11(5):1–26.
- Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. *Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 312–320.
- Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 1–18. Association for Computational Linguistics.
- Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PLOS ONE*, 10(12):1–22.
- Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*. European Language Resources Association.
- Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.
- Richard I. Parker, Kimberly J. Vannest, and John L. Davis. 2011. Effect size in single-case research: A review of nine nonoverlap techniques. *Behavior Modification*, 35(4):303–322.
- Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. The edinburgh twitter corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26. Association for Computational Linguistics.
- Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, pages 43–48. Association for Computational Linguistics.
- Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language Resources and Evaluation*, 47(1):239–268.
- Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. European Language Resources Association.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 73–80. Association for Computational Linguistics and Dublin City University.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. 2012. Harnessing twitter “big data” for automatic emotion identification. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, pages 587–592. IEEE Computer Society.

Deirdre Wilson and Dan Sperber. 2007. On verbal irony. *Irony in language and thought*, pages 35–56.

A Systematic Study of Neural Discourse Models for Implicit Discourse Relation

Attapol T. Rutherford
Yelp
San Francisco, CA, USA
teruth@yelp.com

Vera Demberg
Saarland University
Saarbrücken, Germany
vera@coli.uni-saarland.de

Nianwen Xue
Brandeis University
Waltham, MA, USA
xuen@brandeis.edu

Abstract

Inferring implicit discourse relations in natural language text is the most difficult subtask in discourse parsing. Many neural network models have been proposed to tackle this problem. However, the comparison for this task is not unified, so we could hardly draw clear conclusions about the effectiveness of various architectures. Here, we propose neural network models that are based on feedforward and long-short term memory architecture and systematically study the effects of varying structures. To our surprise, the best-configured feedforward architecture outperforms LSTM-based model in most cases despite thorough tuning. Further, we compare our best feedforward system with competitive convolutional and recurrent networks and find that feedforward can actually be more effective. For the first time for this task, we compile and publish outputs from previous neural and non-neural systems to establish the standard for further comparison.

1 Introduction

The discourse structure of a natural language text has been analyzed and conceptualized under various frameworks (Mann and Thompson, 1988; Lascarides and Asher, 2007; Prasad et al., 2008). The Penn Discourse TreeBank (PDTB) and the Chinese Discourse Treebank (CDTB), currently the largest corpora annotated with discourse structures in English and Chinese respectively, view the discourse structure of a text as a set of discourse relations (Prasad et al., 2008; Zhou and Xue, 2012). Each discourse relation (e.g. causal or temporal) is grounded by a discourse connective (e.g. *because* or *meanwhile*) taking two text segments as argu-

ments (Prasad et al., 2008). Implicit discourse relations are those where discourse connectives are omitted from the text and yet the discourse relations still hold.

While classifying explicit discourse relations is relatively easy, as the discourse connective itself provides a strong cue for the discourse relation (Pitler et al., 2008), the classification of implicit discourse relations has proved to be notoriously hard and remained one of the last missing pieces in an end-to-end discourse parser (Xue et al., 2015). In the absence of explicit discourse connectives, implicit discourse relations have to be inferred from their two arguments. Previous approaches on inferring implicit discourse relations have typically relied on features extracted from their two arguments. These features include the Cartesian products of the word tokens in the two arguments as well as features manually crafted from various lexicons such as verb classes and sentiment lexicons (Pitler et al., 2009; Rutherford and Xue, 2014). These lexicons are used mainly to offset the data sparsity problem created by pairs of word tokens used directly as features.

Neural network models are an attractive alternative for this task, but it is not clear how well they will fare with a small dataset, typically found in discourse annotation projects. Many neural approaches have been proposed. However, we lack a unified standard comparison to really learn whether we make any progress at all because not all past studies agree on the same experimental settings such as label sets to use. Previous work used four binary classification (Pitler et al., 2008; Rutherford and Xue, 2014), 4-way coarse sense classification (Rutherford and Xue, 2015), and intermediate sense classification (Lin et al., 2009). CoNLL Shared Task introduces a unified scheme for evaluation along with a new unseen test set in English in 2015 (Xue et al., 2015) and in Chinese in 2016 (Xue et al., 2016). We want to corrob-

rate this new evaluation scheme by running more benchmark results and providing the output under this evaluation scheme. We systematically compare the relative advantages of different neural architectures and publish the outputs from the systems for the research community to conduct further analysis.

In this work, we explore multiple neural architectures in an attempt to find the best distributed representation and neural network architecture suitable for this task in both English and Chinese. We do this by probing the different points on the spectrum of structurality from structureless bag-of-words models to sequential and tree-structured models. We use feedforward, sequential long short-term memory (LSTM), and tree-structured LSTM models to represent these three points on the spectrum. To the best of our knowledge, there is no prior study that investigates the contribution of the different architectures in neural discourse analysis.

Our main contributions and findings from this work can be summarized as follows:

- We establish that the simplest feedforward discourse model outperforms systems with surface features and perform comparably with or even outperforms recurrent and convolutional architectures. This holds across different label sets in English and in Chinese.
- We investigate the contribution of the linguistic structures in neural discourse modeling and found that high-dimensional word vectors trained on a large corpus can compensate for the lack of structures in the model, given the small amount of annotated data.
- We collect and publish the system outputs from many neural architectures on the standard experimental settings for the community to conduct more error analysis. These are made available on the author’s website.

2 Model Architectures

Following previous work, we assume that the two arguments of an implicit discourse relation are given so that we can focus on predicting the senses of the implicit discourse relations. The input to our model is a pair of text segments called Arg1 and Arg2, and the label is one of the senses defined in the Penn

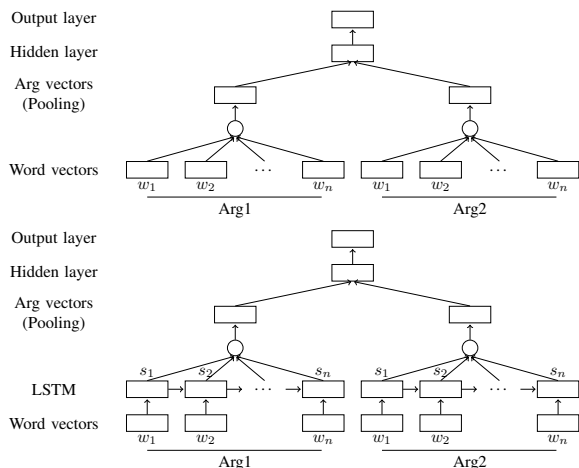


Figure 1: (Top) Feedforward architecture. (Bottom) Sequential Long Short-Term Memory architecture.

Discourse Treebank as in the example below:

Input:

- Arg1 Senator Pete Domenici calls this effort “the first gift of democracy”
 Arg2 The Poles might do better to view it as a Trojan Horse.

Output:

Sense Comparison.Contrast

In all architectures, each word in the argument is represented as a k -dimensional word vector trained on an unannotated data set. We use various model architectures to transform the semantics represented by the word vectors into distributed continuous-valued features. In the rest of the section, we explain the details of the neural network architectures that we design for the implicit discourse relations classification task. The models are summarized schematically in Figure 1.

2.1 Bag-of-words Feedforward Model

This model does not model the structure or word order of a sentence. The features are simply obtained through element-wise pooling functions. Pooling is one of the key techniques in neural network modeling of computer vision (Krizhevsky et al., 2012; LeCun et al., 2010). Max pooling is known to be very effective in vision, but it is unclear what pooling function works well when it comes to pooling word vectors. Summation pooling and mean pooling have been claimed to perform well at composing meaning of a short phrase from individual word vectors (Le and Mikolov,

2014; Blacoe and Lapata, 2012; Mikolov et al., 2013b; Braud and Denis, 2015). The Arg1 vector a^1 and Arg2 vector a^2 are computed by applying element-wise pooling function f on all of the N_1 word vectors in Arg1 $w_{1:N_1}^1$ and all of the N_2 word vectors in Arg2 $w_{1:N_2}^2$ respectively:

$$\begin{aligned} a_i^1 &= f(w_{1:N_1,i}^1) \\ a_i^2 &= f(w_{1:N_2,i}^2) \end{aligned} \qquad \begin{aligned} a_i^1 &= f(s_{1:N_1,i}^1) \\ a_i^2 &= f(s_{1:N_2,i}^2) \end{aligned}$$

We consider three different pooling functions namely max, summation, and mean pooling functions:

$$\begin{aligned} f_{max}(w_{1:N}, i) &= \max_{j=1}^N w_{j,i} \\ f_{sum}(w_{1:N}, i) &= \sum_{j=1}^N w_{j,i} \\ f_{mean}(w_{1:N}, i) &= \sum_{j=1}^N w_{j,i}/N \end{aligned}$$

Inter-argument interaction is modeled directly by the hidden layers that take argument vectors as features. Discourse relations cannot be determined based on the two arguments individually. Instead, the sense of the relation can only be determined when the arguments in a discourse relation are analyzed jointly. The first hidden layer h_1 is the non-linear transformation of the weighted linear combination of the argument vectors:

$$h_1 = \tanh(W_1 \cdot a^1 + W_2 \cdot a^2 + b_{h_1})$$

where W_1 and W_2 are $d \times k$ weight matrices and b_{h_1} is a d -dimensional bias vector. Further hidden layers h_t and the output layer o follow the standard feedforward neural network model.

$$\begin{aligned} h_t &= \tanh(W_{h_t} \cdot h_{t-1} + b_{h_t}) \\ o &= \text{softmax}(W_o \cdot h_T + b_o) \end{aligned}$$

where W_{h_t} is a $d \times d$ weight matrix, b_{h_t} is a d -dimensional bias vector, and T is the number of hidden layers in the network.

2.2 Sequential Long Short-Term Memory (LSTM)

A sequential Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) models the semantics of a sequence of words through the use of hidden state vectors. Therefore, the word ordering does affect the resulting hidden state vectors, unlike the bag-of-word model. For each word vector

at word position t , we compute the corresponding hidden state vector s_t and the memory cell vector from the previous step, using standard formula for LSTM. The argument vectors are the results of applying a pooling function over the hidden state vectors.

$$\begin{aligned} a_i^1 &= f(s_{1:N_1,i}^1) \\ a_i^2 &= f(s_{1:N_2,i}^2) \end{aligned}$$

In addition to the three pooling functions that we describe in the previous subsection, we also consider using only the last hidden state vector, which should theoretically be able to encode the semantics of the entire word sequence.

$$f_{last}(s_{1:N}, i) = s_{N,i}$$

Inter-argument interaction and the output layer are modeled in the same fashion as the bag-of-words model once the argument vector is computed.

2.3 Tree LSTM

The principle of compositionality leads us to believe that the semantics of the argument vector should be determined by the syntactic structures and the meanings of the constituents. For a fair comparison with the sequential model, we apply the same formulation of LSTM on the binarized constituent parse tree. The hidden state vector now corresponds to a constituent in the tree. These hidden state vectors are then used in the same fashion as the sequential LSTM. The mathematical formulation is the same as Tai et al. (2015).

This model is similar to the recursive neural networks proposed by Ji and Eisenstein (2015). Our model differs from their model in several ways. We use the LSTM networks instead of the ‘‘vanilla’’ RNN formula and expect better results due to less complication with vanishing and exploding gradients during training. Furthermore, our purpose is to compare the influence of the model structures. Therefore, we must use LSTM cells in both sequential and tree LSTM models for a fair and meaningful comparison. The more in-depth comparison of our work and recursive neural network model by Ji and Eisenstein (2015) is provided in the discussion section.

3 Corpora and Implementation

The Penn Discourse Treebank (PDTB) We use the PDTB due to its theoretical simplicity in discourse analysis and its reasonably large size. The

Sense	Train	Dev	Test
Comparison.Concession	192	5	5
Comparison.Contrast	1612	82	127
Contingency.Cause	3376	120	197
Contingency.Pragmatic cause	56	2	5
Expansion.Alternative	153	2	15
Expansion.Conjunction	2890	115	116
Expansion.Instantiation	1132	47	69
Expansion.List	337	5	25
Expansion.Restatement	2486	101	190
Temporal.Asynchronous	543	28	12
Temporal.Synchrony	153	8	5
Total	12930	515	766

Table 1: The distribution of the level 2 sense labels in the Penn Discourse Treebank. The instances annotated with two labels are not double-counted, and partial labels are excluded.

annotation is done as another layer on the Penn Treebank on Wall Street Journal sections. Each relation consists of two spans of text that are minimally required to infer the relation, and the sense is organized hierarchically. The classification problem can be formulated in various ways based on the hierarchy. Previous work in this task has been done over three schemes of evaluation: top-level 4-way classification (Pitler et al., 2009), second-level 11-way classification (Lin et al., 2009; Ji and Eisenstein, 2015), and modified second-level classification introduced in the CoNLL 2015 Shared Task (Xue et al., 2015). We focus on the second-level 11-way classification because the labels are fine-grained enough to be useful for downstream tasks and also because the strongest neural network systems are tuned to this formulation. If an instance is annotated with two labels ($\sim 3\%$ of the data), we only use the first label. Partial labels, which constitute $\sim 2\%$ of the data, are excluded. Table 3 shows the distribution of labels in the training set (sections 2-21), development set (section 22), and test set (section 23).

Training Weight initialization is uniform random, following the formula recommended by Bengio (2012). The cost function is the standard cross-entropy loss function, as the hinge loss function (large-margin framework) yields consistently inferior results. We use Adagrad as the optimization algorithm of choice. The learning rates are tuned over a grid search. We monitor the accuracy on the development set to determine convergence and prevent overfitting. L2 regularization and/or dropout do not make a big impact on performance in our case, so we do not use them in the final re-

sults.

Implementation All of the models are implemented in Theano (Bergstra et al., 2010; Bastien et al., 2012). The gradient computation is done with symbolic differentiation, a functionality provided by Theano. Feedforward models and sequential LSTM models are trained on CPUs on Intel Xeon X5690 3.47GHz, using only a single core per model. A tree LSTM model is trained on a GPU on Intel Xeon CPU E5-2660. All models converge within hours.

4 Experiment on the Second-level Sense in the PDTB

We want to test the effectiveness of the inter-argument interaction and the three models described above on the fine-grained discourse relations in English. The data split and the label set are exactly the same as previous works that use this label set (Lin et al., 2009; Ji and Eisenstein, 2015).

Preprocessing All tokenization is taken from the gold standard tokenization in the PTB (Marcus et al., 1993). We use the Berkeley parser to parse all of the data (Petrov et al., 2006). We test the effects of word vector sizes. 50-dimensional and 100-dimensional word vectors are trained on the training sections of WSJ data, which is the same text as the PDTB annotation. Although this seems like too little data, 50-dimensional WSJ-trained word vectors have previously been shown to be the most effective in this task (Ji and Eisenstein, 2015). Additionally, we also test the off-the-shelf word vectors trained on billions of tokens from Google News data freely available with the `word2vec` tool. All word vectors are trained on the Skip-gram architecture (Mikolov et al., 2013b; Mikolov et al., 2013a). Other models such as GloVe and continuous bag-of-words seem to yield broadly similar results (Pennington et al., 2014). We keep the word vectors fixed, instead of fine-tuning during training.

4.1 Results

The feedforward model performs best overall among all of the neural architectures we explore (Table 2). It outperforms the recursive neural network with bilinear output layer introduced by Ji and Eisenstein (2015) ($p < 0.05$; bootstrap test) and performs comparably with the surface feature baseline (Lin et al., 2009), which uses var-

Architecture	k	No hidden layer				1 hidden layer				2 hidden layers			
		max	mean	sum	last	max	mean	sum	last	max	mean	sum	last
Feedforward	50	31.85	31.98	29.24	-	33.28	34.98	37.85	-	34.85	35.5	38.51	-
LSTM	50	31.85	32.11	34.46	31.85	34.07	33.15	36.16	34.34	36.16	35.11	37.2	35.24
Tree LSTM	50	28.59	28.32	30.93	28.72	29.89	30.15	32.5	31.59	32.11	31.2	32.5	29.63
Feedforward	100	33.29	32.77	28.72	-	36.55	35.64	37.21	-	36.55	36.29	37.47	-
LSTM	100	30.54	33.81	35.9	33.02	36.81	34.98	37.33	35.11	37.46	36.68	37.2	35.77
Tree LSTM	100	29.76	28.72	31.72	31.98	31.33	26.89	33.02	33.68	32.63	31.07	32.24	33.02
Feedforward	300	32.51	34.46	35.12	-	35.77	38.25	39.56	-	35.25	38.51	39.03	-
LSTM	300	28.72	34.59	35.24	34.64	38.25	36.42	37.07	35.5	38.38	37.72	37.2	36.29
Tree LSTM	300	28.45	31.59	32.76	26.76	33.81	32.89	33.94	32.63	32.11	32.76	34.07	32.50

Table 3: Compilation of all experimental configurations for 11-way classification on the PDTB test set. k is the word vector size. Bold-faced numbers indicate the best performance for each architecture, which is also shown in Table 2.

Model	Accuracy
<i>PDTB Second-level senses</i>	
Most frequent tag baseline	25.71
Our best tree LSTM	34.07
Ji & Eisenstein, (2015)	36.98
Our best sequential LSTM variant	38.38
Our best feedforward variant	39.56
Lin et al., (2009)	40.20

Table 2: Performance comparison across different models for second-level senses.

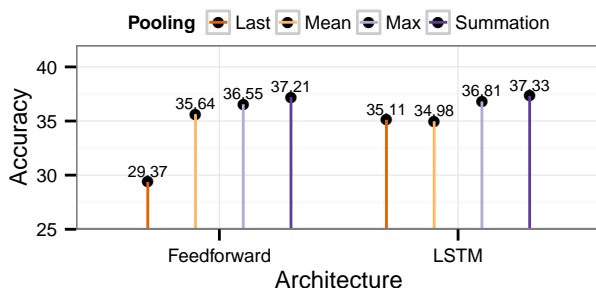


Figure 2: Summation pooling gives the best performance in general. The results are shown for the systems using 100-dimensional word vectors and one hidden layer.

ious lexical and syntactic features and extensive feature selection. Tree LSTM achieves inferior accuracy than our best feedforward model. The best configuration of the feedforward model uses 300-dimensional word vectors, one hidden layer, and the summation pooling function to derive argument feature vectors. The model behaves well during training and converges in less than an hour on a CPU.

The sequential LSTM model outperforms the feedforward model when word vectors are not high-dimensional and not trained on a large cor-

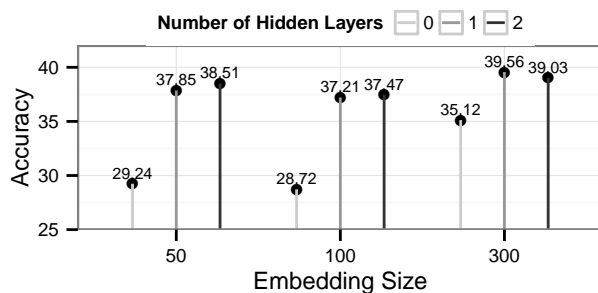


Figure 3: Inter-argument interaction can be modeled effectively with hidden layers. The results are shown for the feedforward models with summation pooling, but this effect can be observed robustly in all architectures we consider.

pus (Figure 4). Moving from 50 units to 100 units trained on the same dataset, we do not observe much of a difference in performance in both architectures, but the sequential LSTM model beats the feedforward model in both settings (Table 3). This suggests that only 50 dimensions are needed for the WSJ corpus. However, the trend reverses when we move to 300-dimensional word vectors trained on a much larger corpus. These results suggest an interaction between the lexical information encoded by word vectors and the structural information encoded by the model itself.

Hidden layers, especially the first one, make a substantial impact on performance. This effect is observed across all architectures (Figure 3). Strikingly, the improvement can be as high as 8% absolute when used with the feedforward model with small word vectors. We tried up to four hidden layers and found that the additional hidden layers yield diminishing—if not negative—returns. These effects are not an artifact of the training process as we have tuned the models quite extensively, although it might be the case that we do not



Figure 4: Comparison between feedforward and sequential LSTM when using summation pooling function.

have sufficient data to fit those extra parameters.

Summation pooling is effective for both feedforward and LSTM models (Figure 2). The word vectors we use have been claimed to have some additive properties (Mikolov et al., 2013b), so summation pooling in this experiment supports this claim. Max pooling is only effective for LSTM, probably because the values in the word vector encode the abstract features of each word relative to each other. It can be trivially shown that if all of the vectors are multiplied by -1, then the results from max pooling will be totally different, but the word similarities remain the same. The memory cells and the state vectors in the LSTM models transform the original word vectors to work well the max pooling operation, but the feedforward net cannot transform the word vectors to work well with max pooling as it is not allowed to change the word vectors themselves.

4.2 Why does the feedforward model outperform the LSTM models?

Summing up vectors indeed works better than recurrent models. We provide further evidence for this claim in Section 5. Sequential and tree LSTM models might work better if we are given larger amount of data. We observe that LSTM models outperform the feedforward model when word vectors are smaller, so it is unlikely that we train the LSTMs incorrectly. It is more likely that we do not have enough annotated data to train a more powerful model such as LSTM. In previous work, LSTMs are applied to tasks with a lot of labeled data compared to mere 12,930 instances that we have (Vinyals et al., 2015; Chiu and Nichols, 2015; Írsoy and Cardie, 2014). Another explanation comes from the fact that the contextual information encoded in the word vectors can compen-

sate for the lack of structure in the model in this task. Word vectors are already trained to encode the words in their linguistic context especially information from word order.

Our discussion would not be complete without explaining our results in relation to the recursive neural network model proposed by Ji and Eisenstein (2015). Why do sequential LSTM models outperform recursive neural networks or tree LSTM models? Although this first comes as a surprise to us, the results are consistent with recent works that use sequential LSTM to encode syntactic information. For example, Vinyals et al. (2015) use sequential LSTM to encode the features for syntactic parse output. Tree LSTM seems to show improvement when there is a need to model long-distance dependency in the data (Tai et al., 2015; Li et al., 2015). Furthermore, the benefits of tree LSTM are not readily apparent for a model that discards the syntactic categories in the intermediate nodes and makes no distinction between heads and their dependents, which are at the core of syntactic representations.

Another point of contrast between our work and Ji and Eisenstein’s (2015) is the modeling choice for inter-argument interaction. Our experimental results show that the hidden layers are an important contributor to the performance for all of our models. We choose linear inter-argument interaction instead of bilinear interaction, and this decision gives us at least two advantages. Linear interaction allows us to stack up hidden layers without the exponential growth in the number of parameters. Secondly, using linear interaction allows us to use high dimensional word vectors, which we found to be another important component for the performance. The recursive model by Ji and Eisenstein (2015) is limited to 50 units due to the bilinear layer. Our choice of linear inter-argument interaction and high-dimensional word vectors turns out to be crucial to building a competitive neural network model for classifying implicit discourse relations.

5 Extending the results across neural architectures, label sets, and languages

We want to provide further evidence that feedforward models perform well without surface features or without sophisticated recurrent or convolutional structures across different label sets and languages as well. Toward that goal, we evaluate

our models on non-explicit discourse relation data used in English and Chinese CoNLL 2016 Shared Task.

5.1 English discourse relations

We follow the experimental setting used in CoNLL 2015-2016 Shared Task. To compare our results against previous systems, we compile all of the official system outputs, and make them publicly available. The label set is modified by the shared task organizers into 15 different senses including EntRel as another sense (Xue et al., 2015; Xue et al., 2016). We use the 300-dimensional word vector used in the previous experiment and tune the number of hidden layers and hidden units on the development set. We consider the following models: Bidirectional-LSTM (Akanksha and Eisenstein, 2016), two flavors of convolutional networks (Qin et al., 2016; Wang and Lan, 2016), two variations of simple argument pooling (Mihaylov and Frank, 2016; Schenk et al., 2016), and the best system using surface features alone (Wang and Lan, 2015). The comparison results and brief system descriptions are shown in Table 4.

Our model presents the state-of-the-art system on the blind test set in English. We once again confirm that manual features are not necessary for this task and that our feedforward network outperforms the best available LSTM and convolutional networks in many settings despite its simplicity. While performing well in-domain, convolutional networks degrade sharply when tested on the blind slightly out-of-domain dataset.

5.2 Chinese discourse relations

We evaluate our model on the Chinese Discourse Treebank (CDTB) because its annotation is the most comparable to the PDTB (Zhou and Xue, 2015). The sense set consists of 10 different senses, which are not organized in a hierarchy, unlike the PDTB. We use the version of the data provided to the CoNLL 2016 Shared Task participants. This version has 16,946 instances of discourse relations total in the combined training and development sets. The test set is not yet available at the time of submission, so the system is evaluated based on the average accuracy over 7-fold cross-validation on the combined set of training and development sets.

To establish baseline comparison, we use MaxEnt models loaded with the feature sets previously shown to be effective for English, namely

Model	Acc.
<i>CoNLL-ST 2015-2016 English (WSJ Test set)</i>	
Most frequent tag baseline	21.36
Our best LSTM variant	31.76
Wang and Lan (2015) - winning team	34.45
Our best feedforward variant	36.13
<i>CoNLL-ST 2016 Chinese (CTB Test set)</i>	
Most frequent tag baseline	77.14
ME + Production rules	80.81
ME + Dependency rules	82.34
ME + Brown pairs (1000 clusters)	82.36
Out best LSTM variant	82.48
ME + Brown pairs (3200 clusters)	82.98
ME + Word pairs	83.13
ME + All feature sets	84.16
Our best feedforward variant	85.45

Table 5: Our best feedforward variant significantly outperforms the systems with surface features ($p < 0.05$). ME=Maximum Entropy model

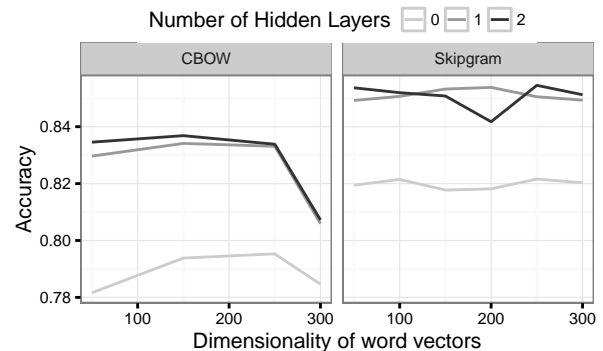


Figure 5: Comparing the accuracies across Chinese word vectors for feedforward model.

dependency rule pairs, production rule pairs (Lin et al., 2009), Brown cluster pairs (Rutherford and Xue, 2014), and word pairs (Marcu and Echihabi, 2002). We use information gain criteria to select the best subset of each feature set, which is crucial in feature-based discourse parsing.

Chinese word vectors are induced through CBOW and Skipgram architecture in `word2vec` (Mikolov et al., 2013a) on Chinese Gigaword corpus (Graff and Chen, 2005) using default settings. The number of dimensions that we try are 50, 100, 150, 200, 250, and 300. We induce 1,000 and 3,000 Brown clusters on the Gigaword corpus.

Table 5 shows the results for the models which are best tuned on the number of hidden units, hidden layers, and the types of word vectors. The feedforward variant of our model significantly outperforms the strong baselines in both English and Chinese ($p < 0.05$ bootstrap test). This suggests that our approach is robust against different label

Systems	Arg vector	Features?	Blind set	WSJ Test	WSJ Dev
Ours	Summing vectors	No	0.3767	0.3613	0.4032
Akanksha & Eisenstein (2016)	2-layer Bi-LSTM	Yes	0.3675	0.3495	0.4072
Qin et al. (2016)	Convolutional net	No	0.3538	0.3820	0.4632
Mihaylov & Frank (2016)	Averaging vectors	Yes	0.3451	0.3919	0.4032
Schenk et al. (2016)	Avg + Product	No	0.3185	0.3761	0.4542
Wang & Lan (2016)	Convolutional net	No	0.3418	0.4091	0.4642
Wang & Lan (2015)	N/A	Yes	0.3629	0.3445	0.4272

Table 4: Comparing various systems on the CoNLL 2016 Shared Task standard datasets. Manual features are no longer needed for a competitive system. While performing well in-domain, convolutional networks degrade sharply when tested on the blind slightly out-of-domain dataset.

sets, and our findings are valid across languages. Our Chinese model outperforms all of the feature sets known to work well in English despite using only word vectors. The choice of neural architecture used for inducing Chinese word vectors turns out to be crucial. Chinese word vectors from Skip-gram model perform consistently better than the ones from CBOW model (Figure 5). These two types of word vectors do not show much difference in the English tasks.

6 Related Work

The prevailing approach for this task is to use surface features derived from various semantic lexicons (Pitler et al., 2009), reducing the number of parameters by mapping raw word tokens in the arguments of discourse relations to a limited number of entries in a semantic lexicon such as polarity and verb classes. Along the same vein, Brown cluster assignments have also been used as a general purpose lexicon that requires no human manual annotation (Rutherford and Xue, 2014). However, these solutions still suffer from the data sparsity problem and almost always require extensive feature selection to work well (Park and Cardie, 2012; Lin et al., 2009; Ji and Eisenstein, 2015). The work we report here explores the use of the expressive power of distributed representations to overcome the data sparsity problem found in the traditional feature engineering paradigm.

Neural network modeling has been explored to some extent in the context of this task. Recently, Braud and Denis (2015) tested various word vectors as features for implicit discourse relation classification and show that distributed features achieve the same level of accuracy as one-hot representations in some experimental settings. Ji et al. (2015; 2016) advance the state of the art for this task by using recursive and recurrent neural networks. In the work we report here, we

systematically explore the use of different neural network architectures and show that when high-dimensional word vectors are used as input, a simple feed-forward architecture can outperform more sophisticated architectures such as sequential and tree-based LSTM networks, given the small amount of data.

Recurrent neural networks, especially LSTM networks, have changed the paradigm of deriving distributed features from a sentence (Hochreiter and Schmidhuber, 1997), but they have not been much explored in the realm of discourse parsing. LSTM models have been notably used to encode the meaning of source language sentence in neural machine translation (Cho et al., 2014; Devlin et al., 2014) and recently used to encode the meaning of an entire sentence to be used as features (Kiros et al., 2015). Many neural architectures have been explored and evaluated, but there is no single technique that is decidedly better across all tasks. The LSTM-based models such as Kiros et al. (2015) perform well across tasks but do not outperform some other strong neural baselines. Ji et al. (2016) uses a joint discourse language model to improve the performance on the coarse-grained label in the PDTB, but in our case, we would like to deduce how well LSTM fares in fine-grained implicit discourse relation classification, which is more practical for application.

7 Conclusions and future work

We report a series of experiments that systematically probe the effectiveness of various neural network architectures for the task of implicit discourse relation classification. We found that a feedforward variant of our model combined with hidden layers and high dimensional word vectors outperforms more complicated LSTM and convolutional models. We also establish that manually crafted surface features are not necessary for

this task. These results hold for different settings and different languages. In addition, we collect and compile the system outputs from all competitive systems and make it available for the research community to conduct further analysis. We encourage that researchers who work on this task to evaluate their systems under the CoNLL Shared Task 2015-2016 scheme to allow for easy comparison and progress tracking.

Acknowledgments

The first author was funded by the German Research Foundation (DFG) as part of SFB 1102: Information Density and Linguistic Encoding

References

- Akanksha and Jacob Eisenstein. 2016. Shallow discourse parsing using distributed argument representations and bayesian optimization. *CoRR*, abs/1606.04503.
- Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.
- Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer.
- James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June. Oral Presentation.
- William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.
- Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Empirical Methods in Natural Language Processing (EMNLP 2015)*.
- Jason P.C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.
- Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1370–1380.
- David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09*, ISBN, 1:58563–58230.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728.
- Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.
- Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- Alex Lascarides and Nicholas Asher. 2007. Segmented discourse representation theory: Dynamic semantics with discourse structure. In *Computing meaning*, pages 87–124. Springer.
- Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.
- Yann LeCun, Koray Kavukcuoglu, and Clément Faret. 2010. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE.

- Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal, September. Association for Computational Linguistics.
- Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics.
- William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.
- Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 368–375. Association for Computational Linguistics.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.
- Todor Mihaylov and Anette Frank. 2016. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, page 100.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.
- Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112. Association for Computational Linguistics.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.
- Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. *Technical Reports (CIS)*, page 884.
- Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 683–691. Association for Computational Linguistics.
- Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltakaki, Livio Robaldo, Aravind K. Joshi, and Bonnie L. Webber. 2008. The penn discourse treebank 2.0. In *LREC*.
- Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. Shallow discourse parsing using convolutional neural network. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, page 70.
- Attapol T. Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, Gothenburg, Sweden, April.
- Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 799–808, Denver, Colorado, May–June. Association for Computational Linguistics.
- Niko Schenk, Christian Chiarcos, Kathrin Donandt, Samuel Rönnqvist, Evgeny A. Stepanov, and Giuseppe Riccardi. 2016. Do we really need all those rich linguistic features? a neural network-based approach to implicit sense labeling. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, page 41.
- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2755–2763. Curran Associates, Inc.
- Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 17–24, Beijing, China, July. Association for Computational Linguistics.
- Jianxiang Wang and Man Lan. 2016. Two end-to-end shallow discourse parsers for english and chinese in conll-2016 shared task. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, page 33.
- Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 1–16, Beijing, China, July. Association for Computational Linguistics.
- Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The conll-2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, Berlin, Germany, August. Association for Computational Linguistics.
- Yuping Zhou and Nianwen Xue. 2012. Pdtb-style discourse annotation of chinese text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 69–77. Association for Computational Linguistics.
- Yuping Zhou and Nianwen Xue. 2015. The chinese discourse treebank: A chinese corpus annotated with discourse relations. *Language Resources and Evaluation*, 49(2):397–431.

Cross-lingual RST Discourse Parsing

Chloé Braud¹, Maximin Coavoux² and Anders Søgaard¹

¹CoAStAL, DIKU, University of Copenhagen, University Park 5, 2100 Copenhagen

²LLF, CNRS, Univ Paris Diderot, Sorbonne Paris Cité

{braud, soegaard}@di.ku.dk

{maximin.coavoux}@etu.univ-paris-diderot.fr

Abstract

Discourse parsing is an integral part of understanding information flow and argumentative structure in documents. Most previous research has focused on inducing and evaluating models from the English RST Discourse Treebank. However, discourse treebanks for other languages exist, including Spanish, German, Basque, Dutch and Brazilian Portuguese. The treebanks share the same underlying linguistic theory, but differ slightly in the way documents are annotated. In this paper, we present (a) a new discourse parser which is simpler, yet competitive (significantly better on 2/3 metrics) to state of the art for English, (b) a harmonization of discourse treebanks across languages, enabling us to present (c) what to the best of our knowledge are the first experiments on cross-lingual discourse parsing.

1 Introduction

Documents can be analyzed as sequences of hierarchical discourse structures. Discourse structures describe the organization of documents in terms of discourse or rhetorical relations. For instance, the three discourse units below can be represented by the tree in Figure 1, where a relation COMPARISON holds between the segments 1 and 2, and a relation CONTRIBUTION links the segment covering the units 1 and 2, and the segment 3.¹

- 1 Consumer spending in Britain rose 0.1% in the third quarter from the second quarter
- 2 and was up 3.8% from a year ago,
- 3 the Central Statistical Office estimated.

¹“NS” and “NN” in Figure 1 describe the nuclearity of the segments, see Section 3.

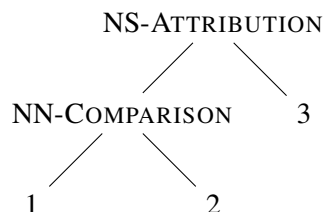


Figure 1: Tree for the structure covering the segments 1 to 3 in document 1384 in the English RST Discourse Treebank.

Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) is a prominent linguistic theory of discourse structures, in which texts are analyzed as constituency trees, such as the one in Figure 1. This theory guided the annotation of the RST Discourse Treebank (RST-DT) (Carlson et al., 2001) for English, from which several text-level discourse parsers have been induced (Hernault et al., 2010; Joty et al., 2012; Feng and Hirst, 2014; Li et al., 2014; Ji and Eisenstein, 2014). Such parsers have proven to be useful for various downstream applications (Daumé III and Marcu, 2009; Burstein et al., 2003; Higgins et al., 2004; Thione et al., 2004; Sporleder and Lapata, 2005; Taboada and Mann, 2006; Louis et al., 2010; Bhattia et al., 2015).

There are discourse treebanks for other languages than English, including Spanish, German, Basque, Dutch, and Brazilian Portuguese. However, most research experimenting with these languages has focused on rule-based systems (Pardo and Nunes, 2008; Maziero et al., 2011) or has been limited to intra-sentential relations (Maziero et al., 2015).

Moreover, all discourse corpora are limited in size, since annotation is complex and time consuming. This data sparsity makes learning hard, especially considering that discourse parsing involves several complex and interacting factors, ranging from syntax and semantics, to pragmat-

ics. We thus propose to harmonize existing corpora in order to leverage information by combining datasets in different languages.

Contributions In this paper, we propose a new discourse parser that is significantly better than existing parsers for English on 2/3 standard metrics. Our parser relies on fewer features than previous work and is arguably algorithmically simpler. Moreover, we present the first end-to-end statistical discourse parsers for other languages than English (6 languages, in total). We also present the first experiments in cross-lingual discourse parsers, showing that discourse parsing is possible even when no or very little labeled data is available for the language of interest. We do so by harmonizing available discourse treebanks, enabling us to apply models across languages. We make the code and preprocessing scripts available for download at <https://bitbucket.org/chloeht/discourse>.

2 Related Work

The first text-level discourse parsers were developed for English, relying mainly on hand-crafted rules and heuristics (Marcu, 2000a; Carlson et al., 2001). Hernault et al. (2010, HILDA) greedily use SVM classifiers to make attachment and labeling decisions, building up a discourse tree. Joty et al. (2012, TSP) build a two-stage parsing system, training separate sequential models (CRF) for the intra- and the inter-sentential levels. These models jointly learn the relation and the structure, and a CKY-like algorithm is used to find the optimal tree. Feng and Hirst (2014) use CRFs only as local models for the inter- and intra-sententials levels. For Brazilian Portuguese, for example, the first system, called DiZer (Pardo and Nunes, 2008; Maziero et al., 2011), was also rule-based, but there has been some work on using classification of intra-sentential relations (Maziero et al., 2015).

Recently studies have focused on building good representations of the data. Feng and Hirst (2012) introduced linguistic features, mostly syntactic and contextual ones. Li et al. (2014) used a recursive neural network that builds a representation for each clause based on the syntactic tree, and then apply two classifiers as in Hernault et al. (2010). This leads to the best performing system for unlabeled structure (85.0 in F_1). The system presented by Ji and Eisenstein (2014, DPLP) jointly learns the representation and the task: a large mar-

gin classifier is used to learn the actions of a shift-reduce parser, optimizing at the same time the loss of the parser and a projection matrix that maps the bag-of-words representation of the discourse units into a new vector space. This system, however, only slightly outperforms the original bag-of-words representation. DPLP is the best performing discourse parser for labeled structure, 71.13 in F_1 for nuclearity and 61.63% for relation.

Our system is similar to these last approaches in learning a representation using a neural network. However, we found that good performance can already be obtained without using all the words in the discourse units, resulting in a parser that is faster and easier to adapt, as demonstrated in our multilingual experiments, see Section 7.

3 RST framework

Discourse analysis In building a discourse structure, the text is first segmented into elementary discourse units (EDU), mostly clauses. EDUs are the smallest discourse units (DUs). Discourse relations are then used to build DUs, recursively. A non-elementary DU is called a complex discourse unit (CDU). The structure of a document is the set of linked DUs. In this paper, we focus on the Rhetorical Structure Theory (RST), a theoretical framework proposed by Mann and Thompson (1988).

Nuclearity A DU is either a *nucleus* or a *satellite*, the nucleus being the most important part of the relation (i.e. of the text), while the satellite contains additional, less important information. In general, this feature depends on the relation: a relation can be either mono-nuclear (with a scheme nucleus-satellite or satellite-nucleus depending on the relative order of the spans), or multi-nuclear. Some relations can be either mono- or multi-nuclear, such as *consequence* or *evaluation* in the RST-DT.

Binary trees In the original RST framework, each relation is associated with an application scheme that defines the nuclearity of the DUs (mono- or multi-nuclear relation), and the number of DUs linked. Among the six schemes, two correspond to a link between more than two DUs, either a nucleus shared between two mono-nuclear relations (e.g. *motivation* and *enablement*) or a relation linking several nuclei (e.g. *list*). Marcu (1997) proposed to simplify the representation to

Corpus	#Doc	#Trees	#Words	#Rel	#Lab	#EDU	max/min/avg	#CDU
En-DT	385	385	206,300	56	110	21,789	304/2/56.6	21,404
Pt-DT	330	329	135,820	32	58	12,573	187/3/38.2	12,244
Es-DT ^a	266	266	69,787	29	43	4,019	77/2/11.5	3,671
De-DT	174	173	32,274	30	46	2,790	24/10/16.1	2,617
Nl-DT	80	80	27,920	31	51	2,345	47/14/29.3	2,265
Eu-DT	88	85	27,982	31	50	2,396	68/3/28.2	2,311

Table 1: Number of documents (#Doc), trees (#Trees, less than #Doc when we were unable to parse a document, see Section 4.2), words (#Words, see Section 6), relations (#Rel, originally), labels (#Lab, relation and nuclearity), EDUs (#EDU, max/min/avg number of EDUs per document), and CDUs (#CDU).

^aThe test set contains 84 documents doubly annotated, we report figures for annotator A.

binary trees, and all discourse parsers are built on a binary representation.

4 Data

We test our discourse parser on six languages, using available RST corpora harmonized as described in Section 4.2. Information about the datasets are summarized in Table 1.

4.1 RST corpora

English The RST Discourse Treebank (Carlson and Marcu, 2001), from now on En-DT, is the most widely used corpus to build discourse parsers. It contains 385 documents in English from the Wall Street Journal. The relation set contains 56 relations (ignoring nuclearity and embedding information²). The inter-annotator agreement scores are 88.70 for the unlabeled structure (score “Span”), 77.72 for the structure with nuclearity (“Nuclearity”) and 65.75 with relations (“Relation”).³

Brazilian Portuguese We merged all the corpora annotated for Brazilian Portuguese, as in (Maziero et al., 2015), to form the Pt-DT. The largest corpus is CST-News⁴ (Cardoso et al., 2011), it is composed of 140 documents from the news domain annotated with 31 relations. Authors report agreement scores corresponding to nuclearity (0.78 in F_1) and relations (0.66).

The other corpora are: Summ-it⁵ (Collovini et al., 2007) – 50 texts from science articles

²In this corpus, the embedded relations are annotated with a specific label (suffix “-e”) that we removed.

³See Section 6 for a description of these metrics.

⁴<http://nilc.icmc.usp.br/CSTNews/login/?next=/CSTNews/>

⁵<http://www.inf.pucrs.br/ontolp/downloads-ontolpplogin.php>

in a newspaper, annotated with 29 relations; Rhetalho⁶ (Pardo and Seno, 2005) – 40 texts from the computer science and news domains, annotated with 23 relations; and CorpusTCC⁶ (Pardo and Nunes, 2003; Pardo and Nunes, 2004) – 100 introductions of scientific texts in computer science, annotated with 31 relations.

Spanish The Spanish RST DT⁷ (da Cunha et al., 2011), from now on Es-DT, contains 267 texts written by specialists on different topics (e.g. astrophysics, economy, law, linguistics) The relation set contains 29 relations. The authors report inter-annotator agreement of 86% in precision for the unlabeled structure, 82.46% for the structure with nuclearity and 76.81% with relations.

German The Postdam Commentary Corpus 2.0⁸ (Stede, 2004; Stede and Neumann, 2014), from now on De-DT, contains newspaper commentaries annotated at several levels. A part of this corpus (MAZ) contains 175 documents annotated within the RST framework using 30 relations.⁹

Dutch The corpus for Dutch (Vliet et al., 2011; Redeker et al., 2012), from now on Nl-DT, contains 80 documents from expository (encyclopedias and science news website) and persuasive (fund-raising letters and commercial advertisements) genres, annotated with 31 relations. The authors report an agreement of 0.83 for discourse spans, 0.77 for nuclearity and 0.70 for relations.

⁶<http://conteudo.icmc.usp.br/pessoas/taspardo/Projects.htm>

⁷http://corpus.iingen.unam.mx/rst/index_en.html

⁸<http://angcl.ling.uni-potsdam.de/resources/pcc.html>

⁹We systematically ignore the first segment of each document, the title, that is not linked to the rest of the text.

Basque The Basque RST DT¹⁰ (Iruskieta et al., 2013), from now on Eu-DT, contains 88 abstracts from three specialized domains – medicine, terminology and science –, annotated with 31 relations. The inter-annotator agreement is 81.67% for the identification of the CDU (Iruskieta et al., 2015), and 61.47% for the identification of the relations.

Other corpora To the best of our knowledge, the only two non English corpora not included are the one annotated for Tamil (Subalalitha and Parthasarathi, 2012) that we were unable to find, and the (intra-sentential) one developed for Chinese (Wu et al., 2016), for which we were unable to produce RST trees since annotation does not contain nuclearity indications.

For English, there are corpora annotated for other domains than the one covered by the En-DT. We however leave out-of-domain evaluation for future work: it requires to decide how to use a corpus annotated only at the sentence level (SFU review corpus)¹¹, or a corpus annotated with genre specific relations (Subba and Di Eugenio, 2009).

4.2 Harmonization of the datasets

Recent discourse parsers built on the En-DT are based on pre-processed data: the corpus contains only binary trees, with the large label set mapped to 18 coarse-grained classes. In this section, we describe this pre-processing step for all corpora used. Discourse corpora have been released under three different file formats: `dis` (En-DT), `lisp` (Rhetalho and CorpusTCC) and `rs3` (all remaining corpora). The first two ones are bracketed format, the third one is an XML encoding. In all cases, the trees encoded do not look like the one in Figure 1: the relations are annotated on the daughter nodes, on the satellite for mono-nuclear relations, or on all the nuclei for multi-nuclear relations. Moreover, in the `rs3` format, the nuclearity of the segments is not directly annotated, it has to be retrieved using the type of the relation (indicated at the beginning of each file) and the previous principle. Our pre-processing step leads to corpora with bracketed files representing directly the RST trees (as in Figure 1) with stand-off annotation of the text of the EDUs.

Note that, even if harmonized, the corpora are not parallel, making it hard to use them to study

language variations for the discourse level. Some preliminary work exists on this question (Iruskieta et al., 2015).

Pre-processing Some documents (format `rs3`) contain several roots or empty segments. We were generally able to remove useless units, that is units that are not linked to other ones within the tree, except for one document in the CST corpus (two roots, both linked to other units).

Another issue concerns unordered EDUs: the structure annotated contains nodes spanning non adjacent EDUs. In general, we were able to correct these cases, but we failed to automatically produce trees spanning only adjacent EDUs for three documents in the Eu-DT, and one document in the De-DT.

Binarization All the corpora contain non-binary trees that we map to binary ones. In the En-DT, common cases of non-binarity are nodes whose daughters all hold the same multi-nuclear relation – indicating that this relation spans multiple DUs, e.g. *list*.¹² In rare cases, the children are two satellites and a nucleus – indicating that the nucleus is shared by the satellites. These configurations are the ones described in (Marcu, 1997) (see Section 3), and choosing right or left-branching leads to a similar interpretation. For the En-DT, right-branching is the chosen strategy since (Soricut and Marcu, 2003).

We found more diverse cases in the other corpora, and, for some of them, right-branching is impossible. It is the case when the daughters are one nucleus (annotated with “Span”, only indicating that this node spans several EDUs) and more than two satellites holding different relations – i.e. the nucleus is shared by all the relations. More precisely, the issue arises when the last two children are satellites. Using right-branching, we end with a node with two satellites as daughters, and thus a ill-formed tree. In order to keep as often as possible the “right-branching by default” strategy, we first do a right-branching and then a left-branching: beginning with four children – S_1-R_i , N_2 -Span, S_3-R_j and S_4-R_k , indicating the relations $R_i(S_1, N_2)$, $R_j(N_2, S_3)$ and $R_k(N_2, S_4)$ ¹³ –, we end up with the tree in Figure 2. Finally, we used a right-branching in all cases, except when the two last children are satellites.

¹⁰<http://ixa2.si.ehu.es/diskurtsosa/en/>

¹¹https://www.sfu.ca/~mtaboada/research/SFU_Review_Corpus.html

¹²Recall that in the original format, the relation is not annotated on the parent node but on the children.

¹³ S being a satellite, N a nucleus and R a relation.

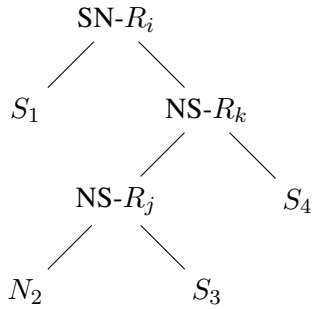


Figure 2: Binary tree for a node X with 4 children: S_1-R_i , N_2 -Span, S_3-R_j and S_4-R_k .

Label set harmonization We map all the relations used in the corpora to the 18 coarse grained classes (Carlson and Marcu, 2001) used to build the most recent discourse parsers on the En-DT.¹⁴

The mapping for the En-DT is given in (Carlson and Marcu, 2001). For all the other corpora, we first map all the relations that exist in this mapping (i.e. used in the En-DT annotation scheme) to their corresponding classes. We end with 18 problematic relations, that is relations that were not used when annotating the En-DT.

Among them, 10 can be mapped easily, because they directly correspond to a class – *explanation* is mapped to the class EXPLANATION, *elaboration* to ELABORATION, *joint* to JOINT –, because they were just renamed – *reformulation* is mapped to the class RESTATEMENT and *solutionhood* (same as *problem-solution*) to TOPIC-COMMENT –, or because they correspond to a more-fine grained formulation of existing relations – *entity-elaboration* is mapped to ELABORATION and the 4 *volitional/non-volitional cause* and *result* are mapped to the class CAUSE, corresponding to the relations *cause* and *result* in the En-DT.

For the remaining relations, we looked at the definition of the relations¹⁵ to decide on a mapping. Note that this label mapping is made quite easy by the fact that all the corpora were annotated following the same underlying theory – they thus use relations defined using similar criteria –, and that we are using a coarse-grained classification – we thus do not need to decide whether a relation is equivalent to another one, but rather whether it fits the properties of the other relations within a specific class. Label mappings for corpora annotated following different frameworks are still

¹⁴The full mapping is provided in Appendix A.

¹⁵<http://www.sfu.ca/rst/01intro/definitions.html>

discussed (Roze, 2013; Benamara and Taboada, 2015).

We decided on the following mapping, considering the properties of the relations and the classes: *parenthetical* – used to give “additional details” – is mapped to ELABORATION, *conjunction* – similar to a *list* with only two elements – to JOINT, *justify* – similar to *Explanation-argumentative* – and *motivation* – quite similar to *reason* and grouped with *evidence* in (Benamara and Taboada, 2015) – to EXPLANATION, *preparation* – presenting preliminary information, increasing the readiness to read the nucleus – to BACKGROUND, and *unconditional* and *unless* – linked to *condition* – to CONDITION.

Finally, note that this mapping does not lead to having the same relation set for all the corpora, and that the relation distribution could vary among the datasets.

5 Discourse Parser

Our discourse parser builds discourse structures from segmented texts, we did not implement discourse segmenters for each language. Discourse segmenters only exist for English (Hernault et al., 2010) (95, 0% in F_1), Brazilian Portuguese (Pardo and Nunes, 2008) (56.8%) and Spanish (da Cunha et al., 2010; da Cunha et al., 2012) (80%). Discourse segmenters can be built quite easily relying only on manual rules as it is the case for the Spanish and Portuguese ones, especially considering that segmentation has generally been made coarser in the corpora built after the En-DT (Vliet et al., 2011). While improving this first step is crucial, we focus on the harder step of tree building.

5.1 Description of the Parser

We used the syntactic parser described in Coavoux and Crabbé (2016), in the static oracle setting. We chose this parser because it can take pre-trained embeddings as input and, more importantly, because it was designed for morphologically rich languages and thus takes as input not only tokens and POS tags, but any token attribute that is then mapped to a real-valued vector, which allows the use of complex features.

The parser is a transition-based constituent parser that uses a lexicalized shift-reduce transition system (Sagae and Lavie, 2005). The transition system is based on two data structures – a *stack* (S) stores partial trees and a *queue* (B) con-

tains the unparsed DUs. A parsing *configuration* is a couple $\langle S, B \rangle$. In the initial configuration, S is empty and B contains the whole document. The parser iteratively applies actions to the current configuration, in order to derive new configurations until it reaches a final state, i.e. a parsing configuration where B is empty and S contains a single element (the root of the tree).

The actions are defined as follows:

- SHIFT pops an EDU from B and pushes it onto S .
- REDUCE-R-X and REDUCE-L-X pop two DUs from S , push a new CDU with the label X on S and assign its nucleus (Left or Right).

Scoring System As in Chen and Manning (2014), at each parsing step, the parser scores actions with a feed-forward neural network. The input of the network is a sequence of typed symbols extracted from the top elements of S and B . The symbols are typically discourse relations or attributes of their nucleus EDU (e.g. first word of EDU, see Section 5.3).

The first layer of the network projects these symbols onto an embedding space (each type of symbol has its own embedding matrix). The following two layers are non-linear layers with a ReLU activation. The output of the network is a probability distribution over possible actions computed by a softmax layer.

To generate a set of training examples $\{a^{(i)}, c^{(i)}\}_{i=1}^N$, we used the static oracle to extract the gold sequence of actions and configurations for each tree in the corpus. The objective function of the parser is the negative log-likelihood of gold actions given corresponding configurations:

$$\mathcal{L}(\theta) = - \sum_{i=1}^N \log P(a^{(i)} | c^{(i)}; \theta)$$

where θ is the set of all parameters, including embedding matrices.

We optimized this objective with the averaged stochastic gradient descent algorithm (Polyak and Juditsky, 1992). At inference time, we used beam-search to find the best-scoring tree.

5.2 Cross-lingual Discourse Parsing

Our first experiments are strictly monolingual, and they are intended to give state-of-the-art performance in a fully supervised setting. We consider

that we need at least 100 documents to build a monolingual model, since we already keep around 65 documents for test and development.

We then evaluate multi-source transfer methods, considering one language as the target and the others as sources. More precisely, we will evaluate two settings: (1) training and optimizing only on the available source data; (2) training on all available data, including target ones if any, and optimizing on the development set of the target language. Setting (1) provides performance when no data are available at all in the target language, while (2) aims at evaluating if one can expect improvements by simply combining all the available data.

When combining the corpora, we cannot ignore lexical information as it has been done for syntactic parsing with delexicalized models (McDonald et al., 2011). Discourse parsing is a semantic task, at least when it comes to predict a rhetorical relation between two spans of text, and information from words have proven to be crucial (Rutherford and Xue, 2014; Braud and Denis, 2015). We thus include word features using bilingual dictionaries – i.e. translating the words used as features into a single language (English) –, or through cross-lingual word embeddings as proposed in (Guo et al., 2015) for dependency parsing. More precisely, we used the cross-lingual word representations presented in (Levy et al., 2017) that allow multi-source learning and have proven useful for POS tagging but also more semantic-oriented tasks, such as dependency parsing and document classification.

5.3 Features

As in previous studies, we used features representing the two EDUs on the top of the stack and the EDU on the queue. If the stack contains CDUs, we use the nuclearity principle to choose the head EDU, converting multi-nuclear relations into nucleus-satellite ones as done since (Sagae, 2009). However, we found that using these information also for the left and right children of the two CDUs on the top of the stack, and adding as a feature the representation built for these two CDUs lead to important improvements.

Lexical features We use the first three words and the last word along with their POS, features that have proven useful for discourse (Pitler et al., 2009), and the words in the *head set* (Sagae, 2009)

– i.e. words whose head in the dependency graph is not in the EDU –, here limited to the first three.¹⁶ This head set contains the head of the sentence (in general, the main event), or words linked to the main clause when the segment does not contain the head (especially, discourse connectives that are subordinating or coordinating conjunctions could be found there). The words at the boundaries could also contain discourse connectives, adverbs or temporal expressions that could be relevant for discourse structure. Note however that these features have been built for English, and they could be less useful for other languages. We leave the question of investigating their utility linked to word order differences for future work.

Note that we do not use all the words in the EDUs as features, contrary to (Li et al., 2014; Ji and Eisenstein, 2014). Our only word features are the words in the head set and at the boundaries, thus 7 words per EDU. When using word embeddings, we concatenate the vectors for each word, each of d dimensions, keeping the same order to build a vector of $7d$ dimensions (e.g., the first word of the EDU corresponds to the first d dimensions, the second has values between d and $2d$).

Position and length Other features are used to represent the position of the EDU in the document and its length in tokens. We use thresholds to distinguish between very long (length $l > 25$ tokens), long ($l > 15$), short ($l > 5$) and very short ($l \leq 5$) EDUs. We also distinguish between the “first” and the “last” EDU in the document, and use also a threshold on the ratio $s = (\text{position of the EDU} / \text{total number of EDUs})$ to separate EDUs at the beginning ($s < 0.25$), in the first middle ($0.25 \leq s < 0.5$), in the second middle ($0.5 \leq s < 0.75$) or in the end ($s \geq 0.75$).

Position of the head We add a boolean feature indicating if the head of the sentence is in the current EDU or outside.

Number/date/percent/money We also use 4 indicators of the presence of a date, a number, an amount of money and a percentage, features that have proven to be useful for discourse (Pitler et al., 2009). We build these features using simple regular expressions.

Corpus	Size dict.	# words	# unk. words
Pt-DT	18,049	13,417	6,929
Es-DT	22,815	6,961	3,231
De-DT	31,900	5,856	1,762
Nl-DT	19,012	3,316	1,428
Eu-DT	1,092	6,553	5,446

Table 2: Dictionary coverage for each dataset on the train set when available, on the dev set else.

6 Experiment settings

Data For the En-DT, we follow previous works in using the official test set of 38 documents. For the Es-DT, we report results on the test set A.¹⁷ For all the other corpora, we randomly choose 38 documents to make a test set, and either use the remaining documents as development set (Nl-DT and Eu-DT), or split them into a development set of 25 documents, the remaining being used as training set (En-DT, Es-DT, Pt-DT and De-DT).

All the results given are based on a gold segmentation of the documents.

Each dataset is parsed using UDPipe,¹⁸ thus tokenizing, splitting into sentences and annotating each document based on the Universal Dependency scheme (Nivre et al., 2016).

The word features for the non-English datasets are translated using available bilingual Wiktionaries¹⁹ without disambiguation, the coverage of each dictionary is given in Table 2. We also look for a translation of the lemma (and of the stems for the languages for which a stemmer²⁰ was available) as a backup strategy. When no translation is found, we keep the original token.

The word embeddings used were built on the EuroParl corpus (Levy et al., 2017). We keep only the 50 first dimensions of the vectors representing the words, our preliminary experiments suggesting no significant differences against keeping the whole 200 dimensions. Unknown words are represented by the average vector of all word vectors. For Basque, we had no access to these embeddings, we thus only report results using bilingual Wiktionaries.

¹⁶Having more than three tokens in the head set is rare.

¹⁷We found similar performance on the other test set.

¹⁸<http://ufal.mff.cuni.cz/udpipe>

¹⁹https://en.wiktionary.org/wiki/User:Matthias_Buchmeier

²⁰<https://pypi.python.org/pypi/snowballstemmer>

Parameter tuning In our experiments we optimized on the development set the following parameters: the learning rate $\in \{0.01, 0.02, 0.03\}$, the learning rate decay constant $\in \{10^{-5}, 10^{-6}, 10^{-7}, 0\}$, the number of iterations $\in [1 - 20]$, and the size of the beam $\in \{1, 2, 4, 8, 16, 32\}$. We fixed the number N of hidden layers to 2 and the size of the hidden layers H to 128 after experimenting on the En-DT (with $N \in \{1, 2, 3\}$ and $H \in \{64, 128\}$).

We fixed the size of the vectors for each feature to 50 for word features,²¹ 16 for POS, 6 for position, 4 for length, and 2 for other features.

Metrics Following (Marcu, 2000b) and most subsequent work, output trees are evaluated against gold trees in terms of how similar they bracket the EDUs (Span), how often they agree about nuclei when predicting a true bracket (Nuclearity), and in terms of the relation label, i.e., the overlap between the shared brackets between predicted and gold trees (Relation).²² These scores are analogous to labeled and unlabeled syntactic parser evaluation metrics.

Baseline Since we do not have state-of-the-art results for most of the languages, we provide results for a simple most frequent baseline (System MFS) that labels all nodes with the most frequent relation in the training or development set – that is NN-JOINT for De-DT and Es-DT, and NS-ELABORATION for the others –, and build the structure by right-branching.

7 Results

Monolingual experiments Monolingual experiments are aimed at evaluating performance for languages having a large annotated corpus (at least 100 documents). Our results are summarized in Table 3. Our parser is competitive with state-of-the-art systems for English (first line in Table 3), with even better performance for unlabeled structure (85.04%) and structure labeled with nuclearity (72.29%). These results show that using all the words in the units (Ji and Eisenstein, 2014; Li et al., 2014), is not as useful as using more contextual information, that is taking more DUs into account (left and right children of the CDUs in the stack). However, the slight drop for Relation shows that

²¹When using embeddings, the final vector is of size 350.

²²We use the evaluation script provided at <https://github.com/jiyfeng/DPLP>.

we probably miss some lexical information, or that we need to choose a more effective combination scheme than concatenation. We plan to use bi-LSTM encoders (Hochreiter and Schmidhuber, 1997) to construct fixed-length representations of EDUs.

For the other languages, performance are still high for unlabeled structure, but far lower for labeled structure except for Spanish. For this language, the quite high performance obtained were unexpected, since the corpus is far much smaller than the Portuguese one. One possible explanation is that the Portuguese corpus is in fact a mix of different corpora, with varied domains, and possibly changes in annotation choices. On the other hand, the low results for German show the sparsity issue since it is the language for which we have the fewest annotations (“#CDU”, see Table 1).

Cross-lingual experiments When only relying on data from different languages (“Cross” in Table 3), we observe a large drop in performance compared to monolingual systems. The source-only discourse parsers still have fairly high performance for unlabeled structure (around 70% or higher), the scores being especially low for relation identification. This could indicate that our representation does not generalize well. But it also comes from differences among the corpora. For example, only the En-DT and the Pt-DT use the relation ATTRIBUTION. This leads to a large drop in performance associated with this relation, when one of these corpora is not in the training data, especially for the source-only system for the En-DT (from 93% in F_1 to 30%). On the other hand, on the En-DT, we observe improvement for other relations either largely represented in all the corpora (e.g. JOINT +3%), or under-represented in the En-DT (e.g. CONDITION +3%).

When combining corpora for source and target languages (“+ dev.” in Table 3), we obtain our best performing system for English, with all scores improved compared to our best monolingual system (+0.8 for Nuclearity and +1.3 for Relation). Otherwise scores are similar to the monolingual case.

Finally, for languages without training set (NI-DT and Eu-DT), this strategy allows us to build parsers outperforming our simple baseline (MFS) by around 11–13% for Span, 8–15% for Nuclearity and 6–11% for Relation. Having at least some annotated data to make a development set allows improvements against only using corpora in other

System	En-DT			Pt-DT			Es-DT			De-DT			NI-DT			Eu-DT		
	Sp	Nuc	Rel	Sp	Nuc	Rel	Sp	Nuc	Rel	Sp	Nuc	Rel	Sp	Nuc	Rel	Sp	Nuc	Rel
MFS	58.2	33.4	22.1	57.3	33.9	23.23	82.0	51.5	17.7	61.3	37.8	13.2	57.9	35.5	22.0	63.2	34.9	18.8
Li et al. ^a	85.0	70.8	58.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DPLP ^a	82.1	71.1	61.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Mono	85.0	72.3	60.1	82.0	65.1	49.9	89.7	72.7	54.4	80.2	53.9	35.0	-	-	-	-	-	-
+ emb.	83.5	68.5	55.9	81.3	62.9	48.8	89.3	72.4	51.4	77.7	51.6	31.1	-	-	-	-	-	-
Cross	76.3	50.5	31.3	76.5	54.6	35.5	78.1	45.4	27.0	76.0	46.0	26.1	69.5	42.1	25.3	78.6	53.0	26.4
+ dev.	85.1	73.1	61.4	81.9	65.1	49.8	88.8	68.0	50.4	79.6	53.6	34.1	69.2	43.4	28.3	76.7	50.5	29.5
Human ^b	88.7	77.7	65.8	-	78	66	86	82.5	76.8	-	-	-	83	77	70	81.7	-	61.5

Table 3: Performance of our monolingual and cross-lingual systems for Span (Sp), Nuclearity (Nuc) and Relation (Rel). “MFS” corresponds to the baseline system described in Section 6; “+ emb.” is the monolingual system using word embeddings; “+dev.” means that the system is optimized on the development set of the target language (vs the union of the source development sets). For cross-lingual systems, we only report our best results using either word embeddings or bilingual dictionaries.

^aScores reported from (Li et al., 2014), and DPLP (Ji and Eisenstein, 2014).

^bFor Brazilian Portuguese, inter-annotator agreement scores are only available for the CST-news corpus ; For Spanish, only precision scores are reported ; For Basque, the scores reported are different (Iruskieta et al., 2015).

languages (around +3% for the NI-DT and the Eu-DT for Relation). On the other hand, we probably overfit our development data for the Eu-DT, since better results were obtained for unlabeled structure (+2%) and structure with nuclearity (+2.5%) using only data in other languages.

Word embeddings Using word embeddings (“+emb” in Table 3) for monolingual systems often leads to an important drop in performance, especially for Relation (from -1.1 to -4.2%). This demonstrates that these embeddings do not provide the large range of information needed for relation identification, a task inherently semantic. We believe however that the results are not too low to prevent for interesting applications. It is noteworthy that the English parser with embeddings is still better than the systems proposed in (Hernault et al., 2010; Joty et al., 2013).

For cross-lingual experiments, the bilingual dictionaries perform generally better than embeddings (except for Pt-DT and De-DT for source-only systems), demonstrating again that we need representations more tailored to the task to leverage all relevant lexical information.

8 Conclusion

We introduced a new discourse parser that obtains state-of-the-art performance for English. We harmonized discourse treebanks for several languages, enabling us to present results for five other languages for which available corpora are smaller, including the first cross-lingual discourse parsing

results in the literature.

Acknowledgements

We thank the three anonymous reviewers for their comments. Chloé Braud and Anders Søgaard were funded by the ERC Starting Grant LOWLANDS No. 313695.

References

- Farah Benamara and Maite Taboada. 2015. Mapping different rhetorical relation annotations: A proposal. In *Proceedings of Starsem*.
- Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of EMNLP*.
- Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of EMNLP*.
- Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the write stuff: automatic identification of discourse structure in student essays. *IEEE Intelligent Systems: Special Issue on Advances in Natural Language Processing*, 18.
- Paula C.F. Cardoso, Erick G. Maziero, Mara Luca Castro Jorge, Eloize R.M. Seno, Ariani Di Felippo, Lucia Helena Machado Rino, Maria das Gracas Volpe Nunes, and Thiago A. S. Pardo. 2011. CSTNews - a discourse-annotated corpus for single and multi-document summarization of news texts in Brazilian Portuguese. In *Proceedings of the 3rd RST Brazilian Meeting*, pages 88–105.

- Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual. Technical report, University of Southern California Information Sciences Institute.
- Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.
- Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182, Berlin, Germany, August. Association for Computational Linguistics.
- Sandra Collovini, Thiago I Carbonel, Juliana Thiesen Fuchs, Jorge César Coelho, Lúcia Rino, and Renata Vieira. 2007. Summ-it: Um corpus anotado com informações discursivas visando a sumarização automática. *Proceedings of TIL*.
- Iria da Cunha, Eric SanJuan, Juan-Manuel Torres-Moreno, Marina Lloberas, and Irene Castellón. 2010. DiSeg: Un segmentador discursivo automático para el español. *Procesamiento del lenguaje natural*, 45:145–152.
- Iria da Cunha, Juan-Manuel Torres-Moreno, and Gerardo Sierra. 2011. On the development of the RST Spanish Treebank. In *Proceedings of the Fifth Linguistic Annotation Workshop, LAW*.
- Iria da Cunha, Eric SanJuan, Juan-Manuel Torres-Moreno, Marina Lloberas, and Irene Castellón. 2012. DiSeg 1.0: The first system for Spanish discourse segmentation. *Expert Syst. Appl.*, 39(2):1671–1678.
- Hal Daumé III and Daniel Marcu. 2009. A noisy-channel model for document compression. In *Proceedings of ACL*.
- Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of ACL*.
- Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of ACL*.
- Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL-IJCNLP*.
- Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1:1–33.
- Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *Proceedings of HLT-NAACL*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Mikel Iruskieta, María J. Aranzabe, Arantza Diaz de Ilarraza, Itziar Gonzalez-Dios, Mikel Lersundi, and Oier Lopez de la Calle. 2013. The RST Basque Treebank: an online search interface to check rhetorical relations. In *Proceedings of the 4th Workshop RST and Discourse Studies*.
- Mikel Iruskieta, Iria da Cunha, and Maite Taboada. 2015. A qualitative comparison method for rhetorical structures: identifying different discourse structures in multilingual corpora. In *Proceedings of LREC*.
- Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of ACL*.
- Shafiq R. Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of EMNLP*.
- Shafiq R. Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of ACL*.
- Omer Levy, Anders Søgaard, and Yoav Goldberg. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of EACL*.
- Jiwei Li, Rumeng Li, and Eduard H. Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of EMNLP*.
- Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of SIGDIAL*.
- William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8:243–281.
- Daniel Marcu. 1997. From discourse structures to text summaries. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 82–88.
- Daniel Marcu. 2000a. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational Linguistics*.
- Daniel Marcu. 2000b. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

- Erick G. Maziero, Thiago A. S. Pardo, Iria da Cunha, Juan-Manuel Torres-Moreno, and Eric SanJuan. 2011. DiZer 2.0-an adaptable on-line discourse parser. In *Proceedings of 3rd RST Brazilian Meeting*, pages 1–17.
- Erick G. Maziero, Graeme Hirst, and Thiago A. S. Pardo. 2015. Adaptation of discourse parsing models for Portuguese language. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS)*.
- Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*.
- Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Çar Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Drozanova, Tomaz Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gokirmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Simon Krek, Veronika Laippala, Lucia Lam, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Măranduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cené-Augusto Pérez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkálnia, Prokopis Prokopidis, Tina Puolalainen, Sampo Pyysalo, Loganathan Ramasamy, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uriá, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jing Xian Wang, Jonathan North Washington, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.
- Thiago A. S. Pardo and Maria das Graças Volpe Nunes. 2003. A construção de um corpus de textos científicos em Português do Brasil e sua marcação retórica. Technical report, Technical Report.
- Thiago A. S. Pardo and Maria das Graças Volpe Nunes. 2004. Relações retóricas e seus marcadores superficiais: Análise de um corpus de textos científicos em Português do Brasil. *Relatório Técnico NILC*.
- Thiago A. S. Pardo and Maria das Graças Volpe Nunes. 2008. On the development and evaluation of a Brazilian Portuguese discourse parser. *Revista de Informática Teórica e Aplicada*, 15(2):43–64.
- Thiago A. S. Pardo and Eloize R. M. Seno. 2005. Rhetalho: Um corpus de referência anotado retoricamente. In *Proceedings of Encontro de Corpora*.
- Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of ACL-IJCNLP*.
- Boris T. Polyak and Anatoli B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, July.
- Gisela Redeker, Ildik Berzlnovich, Nynke van der Vliet, Gosse Bouma, and Markus Egg. 2012. Multi-layer discourse annotation of a dutch text corpus. In *Proceedings of LREC*.
- Charlotte Roze. 2013. *Vers une algèbre des relations de discours*. Ph.D. thesis, Université Paris-Diderot.
- Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of EACL*.
- Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132. Association for Computational Linguistics.
- Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of IWPT 2009*.
- Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL*.
- Caroline Sporleder and Mirella Lapata. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of HLT/EMNLP*.
- Manfred Stede and Arne Neumann. 2014. Potsdam commentary corpus 2.0: Annotation for discourse research. In *Proceedings of LREC*.
- Manfred Stede. 2004. The potsdam commentary corpus. In *Proceedings of the ACL Workshop on Discourse Annotation*.

- C N Subalalitha and Ranjani Parthasarathi. 2012. An approach to discourse parsing using sangati and Rhetorical Structure Theory. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012)*.
- Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of ACL-HLT*.
- Maite Taboada and William C. Mann. 2006. Applications of rhetorical structure theory. *Discourse Studies*, 8:567–588.
- Gian Lorenzo Thione, Martin Van den Berg, Livia Polanyi, and Chris Culy. 2004. Hybrid text summarization: Combining external relevance measures with structural analysis. In *Proceedings of the ACL Workshop Text Summarization Branches Out*.
- Nynke Van Der Vliet, Ildikó Berzlnovich, Gosse Bouma, Markus Egg, and Gisela Redeker. 2011. Building a discourse-annotated Dutch text corpus. In *S. Dipper and H. Zinsmeister (Eds.), Beyond Semantics, Bochumer Linguistische Arbeitsberichte 3*, pages 157–171.
- Yunfang Wu, Fuqiang Wan, Yifeng Xu, and Xueqiang Lü. 2016. A new ranking method for Chinese discourse tree building. *Acta Scientiarum Naturalium Universitatis Pekinensis*, 52(1):65–74.

A Mapping of the relations

Classe	Relations
ATTRIBUTION	<i>attribution, attribution-negative</i>
BACKGROUND	<i>background, circumstance, circunstancia, fondo, preparación, preparation, prestatzea, testuingurua, zirkunstantzia</i>
CAUSE	<i>causa, cause, cause-result, consequence, kausa, non-volitional-cause, non-volitional-result, ondorioa, result, resultado, volitional-cause, volitional-result</i>
COMPARISON	<i>analogy, comparison, preference, proportion</i>
CONDITION	<i>alderantzizko-baldintza, alternativa, aukera, baldintza, condición, condición-inversa, condition, contingency, ez-baldintzatzailea, hypothetical, otherwise, unconditional, unless</i>
CONTRAST	<i>antitesia, antithesis, antítesis, concesión, concession, contrast, contraste, kontrastea, kontzesioa</i>
ELABORATION	<i>definition, e-elaboration, elaboración, elaboration, elaboration-additional, elaboration-general-specific, elaboration-object-attribute, elaboration-part-whole, elaboration-process-step, elaboration-set-member, elaborazioa, example, parenthetical</i>
ENABLEMENT	<i>ahalbideratzea, capacitación, enablement, helburua, propósito, purpose</i>
EVALUATION	<i>comment, conclusion, ebaluazioa, evaluación, evaluation, interpretación, interpretation, interpretazioa</i>
EXPLANATION	<i>ebidentzia, evidence, evidencia, explanation, explanation-argumentative, justificación, justifikazioa, justify, motibazioa, motivación, motivation, reason</i>
JOINT	<i>bateratzea, conjunción, conjunction, disjunction, disjuntzioa, disyunción, joint, konjuntzioa, list, lista, unión</i>
MANNER-MEANS	<i>manner, means, medio, metodoa</i>
SAME-UNIT	<i>same-unit</i>
SUMMARY	<i>birformulazioa, definitu-gabeko-erlazioa, laburpena, reformulación, restatement, resumen, summary</i>
TEMPORAL	<i>inverted-sequence, secuencia, sekuentzia, sequence, temporal-after, temporal-before, temporal-same-time</i>
TEXTUAL-ORGANIZATION	<i>textual-organization</i>
TOPIC-CHANGE	<i>topic-drift, topic-shift</i>
TOPIC-COMMENT	<i>arazo-soluzioa, comment-topic, problem-solution, question-answer, rhetorical-question, solución, solutionhood, statement-response, topic-comment</i>

Table 4: Mapping of all the relations found in the datasets: for each class, we give the set of relation names as they appear in the corpora (removing only the possible suffixes “-e”, “-s”, “-mn”). We ignore the simplest differences in names (e.g. *textual-organization* and *textualorganization*).

Dialog state tracking, a machine reading approach using Memory Network

Julien Perez

Xerox Research Centre Europe
Grenoble, France

julien.perez@xrce.xerox.com

Fei Liu *

The University of Melbourne
Victoria, Australia

fliu3@student.unimelb.edu.au

Abstract

In an end-to-end dialog system, the aim of dialog state tracking is to accurately estimate a compact representation of the current dialog status from a sequence of noisy observations produced by the speech recognition and the natural language understanding modules. This paper introduces a novel method of dialog state tracking based on the general paradigm of machine reading and proposes to solve it using an End-to-End Memory Network, MemN2N, a memory-enhanced neural network architecture. We evaluate the proposed approach on the second Dialog State Tracking Challenge (DSTC-2) dataset. The corpus has been converted for the occasion in order to frame the hidden state variable inference as a question-answering task based on a sequence of utterances extracted from a dialog. We show that the proposed tracker gives encouraging results. Then, we propose to extend the DSTC-2 dataset and the definition of this dialog state task with specific reasoning capabilities like counting, list maintenance, yes-no question answering and indefinite knowledge management. Finally, we present encouraging results using our proposed MemN2N based tracking model.

1 Introduction

One of the core components of state-of-the-art and industrially deployed dialog systems is a dialog state tracker. Its purpose is to provide a compact representation of a dialog produced from past user inputs and system outputs which is called the dialog state. The dialog state summarizes the infor-

mation needed to successfully maintain and finish a dialog, such as users' goals or requests. In the simplest case of a so-called *slot-filling schema*, the state is composed of a predefined set of variables with a predefined domain of expression for each of them. As a matter of fact, in the recent context of end-to-end trainable machine learnt dialog systems, state tracking remains a central element of such architectures (Wen et al., 2016). Current models, mainly based on the principle of discriminative learning, tend to share three common limitations. First, the tracking task is performed using a fixed window of the past dialog utterances as support for decision. Second, the possible correlations between the set of tracked variables are not leveraged and individual trackers tend to be learnt independently. Third, the tracking task is summarized as the capability of inferring values for a predefined set of latent variables. Starting from these observations, we propose to formalize the task of state tracking as a particular instance of machine reading problem. Indeed, these formalization and the proposed resolution model called MemN2N (Weston et al., 2015) allow to define a tracker that is able to decide at the utterance level on the basis of the current entire dialog. Indeed, the model learns to focus its attention on the meaningful parts of the dialog regarding the currently asked slot and can eventually capture possible correlation between slots. As far as our knowledge goes, it is the first attempt to explicitly frame the task of dialog state tracking as a machine reading problem. Finally, such formalization allows for the implementation of approximate reasoning capability that has been shown to be crucial for any machine reading tasks (Weston et al., 2015) while extending the task from slot instantiation to question answering. This paper is structured as follows, Section 2 recalls the main definitions associated to transactional dialogs and describes

*Work carried out as an intern at XRCE

the associated problem of statistical dialog state tracking with both the generative and discriminative approaches. At the end of this section, the limitations of the current models in terms of necessary annotations and reasoning capabilities are addressed. Then, Section 3 depicts the proposed machine reading model for dialog state tracking and proposes to extend a state of the art dialog state tracking dataset, *DSTC-2*, to several simple reasoning capabilities. Section 4 illustrates the approach with experimental results obtained using a state of the art benchmark for dialog state tracking.

2 Dialog state tracking

2.1 Main Definitions

A dialog state tracking task is formalized as follows: at each turn of a dyadic dialog, the dialog agent chooses a dialog act d to express and the user answers with an utterance u . In the simplest case, the dialog state at each turn is defined as a distribution over a set of predefined variables, which define the structure of the state (Williams et al., 2005). This classic state structure is commonly called *slot filling* or *semantic frame*. In this context, the state tracking task consists of estimating the value of a set of predefined variables in order to perform a procedure or transaction which is the purpose of the dialog. Typically, a natural language understanding module processes the user utterance and generates an N-best list $o = \{(d_1, f_1), \dots, (d_n, f_n)\}$, where d_i is the hypothesized user dialog act and f_i is its confidence score. Various approaches have been proposed to define dialog state trackers. The traditional methods used in most commercial implementations use hand-crafted rules that typically rely on the most likely result from an NLU module (Yeh et al., 2014) and hardly models uncertainty. However, these rule-based systems are prone to frequent errors as the most likely result is not always the correct one (Williams, 2014).

More recent methods employ statistical approaches to estimate the posterior distribution over the dialog states allowing them to leverage the uncertainty of the results of the NLU module. In the simplest case where no ASR and NLU modules are employed, as in a text based dialog system (Henderson et al., 2013), the utterance is taken as the observation using a so-called bag of words representation. If an NLU module is available, stan-

dardized dialog act schemes can be considered as observations (Bunt et al., 2010). Furthermore, if prosodic information is available from the ASR component of the dialog system (Milone and Rubio, 2003), it can also be considered as part of the observation definition. A statistical dialog state tracker maintains, at each discrete time step t , the probability distribution over states, $b(s_t)$, which is the system’s *belief* over the state. The actual slot filling process is composed of the cyclic tasks of *information gathering* and integration, in other words – *dialog state tracking*. In such framework, the purpose is to estimate as early as possible in the course of a given dialog the correct instantiation of each variable. In the following, we will assume the state is represented as a set of variables with a set of known possible values associated to each of them. Furthermore, in the context of this paper, only the bag of words has been considered as an observation at a given turn but dialog acts or detected named entity provided by an SLU module could have also been incorporated.

Two statistical approaches have been considered for maintaining the distribution over a state given sequential NLU output. First, the discriminative approach aims to model the posterior probability distribution of the state at time $t + 1$ with regard to state at time t and observations $z_{1:t}$. Second, the generative approach attempts to model the transition probability and the observation probability in order to exploit possible interdependencies between hidden variables that comprise the dialog state.

2.2 Generative Dialog State Tracking

A generative approach to dialog state tracking computes the belief over the state using Bayes’ rule, using the belief from the last turn $b(s_{t-1})$ as a prior and the likelihood given the user utterance hypotheses $p(z_t|s_t)$, with z_t the observation gathered at time t . In prior works (Williams et al., 2005), the likelihood is factored and some independence assumptions are made: $b_t \propto \sum_{s_{t-1}, z_t} p(s_t|z_t, s_{t-1})p(z_t|s_{t-1})b(s_{t-1})$. A typical generative model uses a factorial hidden Markov model (Ghahramani and Jordan, 1997). In this family of approaches, scalability is considered as one of the main issues. One way to reduce the amount of computation is to group the states into partitions, as proposed in the Hidden Information State (HIS) model (Gasic and Young, 2011). Other

approaches to cope with the scalability problem in dialog state tracking is to adopt a factored dynamic Bayesian network by making conditional independence assumptions among dialog state components, and then using approximate inference algorithms such as loopy belief propagation (Thomson and Young, 2010) or a blocked Gibbs sampling as (Raux and Ma, 2011). To cope with such limitations, discriminative methods of state tracking presented in the next part of this section aim at directly model the posterior distribution of the tracked state using a chosen parametric form.

2.3 Discriminative Dialog State Tracking

The discriminative approach of dialog state tracking computes the belief over a state via a parametric model that directly represents the belief $b(s_{t+1}) = p(s_{t+1}|s_t, z_t)$. For example, Maximum Entropy has been widely used in the discriminative approach (Metallinou et al., 2013). It formulates the belief as follows: $b(s) = P(s|x) = \eta \cdot e^{w^T \phi(x,s)}$, where η is the normalizing constant, $x = (d_1^u, d_1^m, s_1, \dots, d_t^u, d_t^m, s_t)$ is the history of user dialog acts, $d_i^u, i \in \{1, \dots, t\}$, the system dialog acts, $d_i^m, i \in \{1, \dots, t\}$, and the sequence of states leading to the current dialog turn at time t . Then, $\phi(\cdot)$ is a vector of feature functions on x and s . Finally, w is the set of model parameters to be learned from annotated dialog data. Finally, deep neural models, performing on a sliding window of features extracted from previous user turns, have also been proposed in (Henderson et al., 2014c; Mrksic et al., 2016). Of the current literature, this family of approaches have proven to be the most efficient for publicly available state tracking datasets. Recently, deep learning based models implementing this strategy (Mrksic et al., 2016; Henderson et al., 2014a; Williams et al., 2016) have shown state of the art results. This approaches tends to leverage unsupervised training word representation (Mikolov et al., 2013).

2.4 Current Limitations

Using error analysis (Henderson et al., 2014b), three limitations can be observed in the application of these inference approaches. First, current models tend to fail at considering long-tail dependencies that occurs on dialogs. For example, coreferences, inter-utterances informations and correlations between slots have been shown to be difficult to handle even with the usage of recurrent network models (Henderson et al., 2014a). To illustrate the

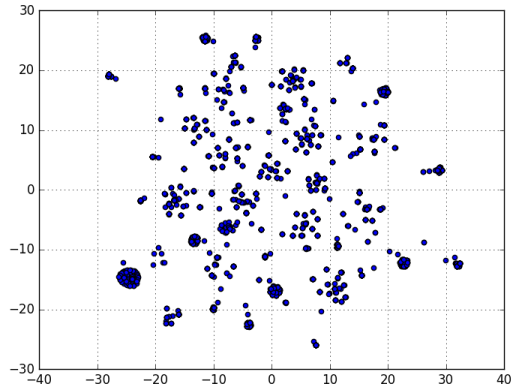


Figure 1: T-SNE transformation of the final state of DSTC-2 train set.

inter-slot correlation, Figure 1 depicted the t-SNE (van der Maaten and Hinton, 2008) projected final state of the dialog of the DSTC-2 training set. On the other hand, reasoning capabilities, as required in machine reading applications (Poon and Domingos, 2010; Etzioni et al., 2007; Berant et al., 2014; Weston et al., 2015) remain absent in these classic formalizations of dialog state tracking. Finally, tracking definition is limited to the capability to instantiate a predefined set of slots. In the next section, we present a model of dialog state tracking that aims at leveraging the current advances of MemN2N, a memory-enhanced neural networks and their approximate reasoning capabilities that seems particularly adapted to the sequential, long range dependency equipped and sparse nature of complex dialog state tracking tasks. Furthermore, this model allows to relax the hypothesis of strict utterance-level annotation that does not corresponds to common practices in industrial applications of transactional conversational user interfaces where annotations tend to be placed at a multi-utterance level or full-dialog level only.

3 Machine Reading Formulation of Dialog State Tracking

We propose to formalize the dialog state tracking task as a machine reading problem (Etzioni et al., 2007; Berant et al., 2014). In this section, we recall the main definitions of the task of machine reading, then describes the MemN2N, a memory-enhanced neural network architectures proposed to handle such tasks in the context of dialogs. Finally, we formalize the task of dialog state tracking as a machine reading problem and propose to

solve it using a memory-enhanced neural architecture of inference.

3.1 Machine Reading

The task of textual understanding has recently been formulated as a supervised learning problem (Kumar et al., 2015; Hermann et al., 2015). This task consists in estimating the conditional probability $p(a|d, q)$ of an answer a to a question q where d is a document. Such an approach requires a large training corpus of {Document - Query - Answer} triples and until now such corpora have been limited to hundreds of examples (Richardson et al., 2013). In the context of dialog state tracking, it can be understood as the capability of inferring a set of latent values l associated with a set of variables v related to a given dyadic or multi-party conversation d , from direct correlation and/or reasoning, using the course of exchanges of utterances, $p(l|d, v)$.

State updates at an utterance-level are rarely provided off-the-shelf from a production environment. In these environments, annotation is often performed afterward for the purpose of logging, monitoring or quality assessment. In the limit cases, as in human-to-human dialog systems, dialog-level annotations remains a common practice of annotation especially in personal assistance, customer care dialogs and, in a more general sense, industrial application of transactional conversational user interfaces. Another frequent setting consist of informing the state after a given number of utterance exchange between the locutors. So an additional effort of specific annotation is often needed in order to train a state of the art statistical state tracking model (Henderson et al., 2014b). In that sense, formalizing dialog state tracking at a sub-dialog level in order to infer hidden state variables with respect to a list of utterances started from the first one to any given utterance of a given dialog seems particularly appropriate. In the context of dialog state tracking challenges, the *DSTC-4* dialog corpus have been designed in such purpose but only consists of 22 dialogs. Concerning the *DSTC-2* corpus, the training data contains 2207 dialogs (15611 turns) and the test set consists of 1117 dialogs (Williams et al., 2016). This dataset is more suitable for our experiments.

For these reasons, the machine reading paradigm becomes a promising formulation for

the general problem of dialog state tracking. Furthermore, current approaches and available datasets for state tracking do not explicitly cover reasoning capabilities such as temporal and spatial reasoning, counting, sorting and deduction. We suggest that in the future dataset dialogs expressing such specific abilities should be developed. In this last part, several reasoning enhancements are suggested to the *DSTC-2* dataset.

3.2 End-to-End Memory Networks

The MemN2N architecture, introduced by (Weston et al., 2015), consists of two main components: supporting memories and final answer prediction. Supporting memories are in turn comprised of a set of input and output memory representations with memory cells. The input and output memory cells, denoted by m_i and c_i , are obtained by transforming the input context x_1, \dots, x_n (i.e a set of utterances) using two embedding matrices A and C (both of size $d \times |V|$ where d is the embedding size and $|V|$ the vocabulary size) such that $m_i = A\Phi(x_i)$ and $c_i = C\Phi(x_i)$ where $\Phi(\cdot)$ is a function that maps the input into a bag of dimension $|V|$.

Similarly, the question q is encoded using another embedding matrix $B \in \mathbb{R}^{d \times |V|}$, resulting in a question embedding $u = B\Phi(q)$. The input memories $\{m_i\}$, together with the embedding of the question u , are utilized to determine the relevance of each of the stories in the context, yielding in a vector of attention weights

$$p_i = \text{softmax}(u^\top m_i) \quad (1)$$

where $\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_i e^{a_i}}$. Subsequently, the response o from the output memory is constructed by the weighted sum:

$$o = \sum_i p_i c_i \quad (2)$$

Other models of parametric encoding for the question and the document have been proposed in (Kumar et al., 2015). For the purpose of this presentation, we will keep with definition of Φ .

For more difficult tasks requiring multiple supporting memories, the model can be extended to include more than one set of input/output memories by stacking a number of memory layers. In this setting, each memory layer is named a hop and the $(k+1)^{\text{th}}$ hop takes as input the output of the k^{th} hop:

$$u^{k+1} = o^k + u^k \quad (3)$$

Lastly, the final step, the prediction of the answer to the question q , is performed by

$$\hat{a} = \text{softmax}(W(o^K + u^K)) \quad (4)$$

where \hat{a} is the predicted answer distribution, $W \in \mathbb{R}^{|V| \times d}$ is a parameter matrix for the model to learn and K the total number of hops.

Two weight tying schemes of the embedding matrices have been introduced in (Weston et al., 2015):

1. **Adjacent:** the output embedding matrix in the k^{th} hop is shared with the input embedding matrix in the $(k+1)^{\text{th}}$ hop, i.e., $A^{k+1} = C^k$ for $k \in \{1, K-1\}$. Also, the weight matrix W in Equation (4) is shared with the output embedding matrix in the last memory hop such that $W^\top = C^K$.
2. **Layer-wise:** all the weight matrices A^k and C^k are shared across different hops, i.e., $A^1 = A^2 = \dots = A^K$ and $C^1 = C^2 = \dots = C^K$.

In the next section, we show how the task of dialog state tracking can be formalized as machine reading task and solved using such memory enhanced model.

3.3 Dialog Reading Model for State Tracking

In this section, we formalize dialog state tracking using the paradigm of machine reading. As far as our knowledge goes, it is the first attempt to apply this approach and develop a specific dataset format, detailed in Section 4, from an existing and publicly available dialog state tracking challenge dataset to fulfill this task. Assuming (1) a dyadic dialog d composed of a list of utterances, (2) a state composed with (2a) a set of variables v_i with $i = \{1, \dots, n\}$ and (2b) a set of corresponding assigned values l_i . One can define a question q_v that corresponds to the specific querying of a variable in the context of a dialog $p(l_i|q_v, d)$. In such context, a dialog state tracking task consists in determining for each variable v , $l^* = \arg \max_{l_i \in L} p(l_i|q_v, d)$, with L the specific domain of expression of a variable v_i .

In addition to the actual dataset, we propose to investigate four general reasoning tasks using *DSTC-2* dataset as a starting point. In such way, we leverage the dataset of *DSTC-2* to create more complex reasoning task than the ones present in the original dialogs of the dataset by performing rule-based modification over the corpus. Obviously, the goal is to develop resolution algorithms

that are not dedicated to a specific reasoning task but inference models that will be as generic as possible. In the rest of the section, each of the reasoning tasks associated with dialog state tracking are described and the generation protocol is explained with examples.

Factoid Questions : This first task corresponds to the current formulation of dialog state tracking. It consists of questions where a previously given a set of supporting facts, potentially amongst a set of other irrelevant facts, provides the answer. This kind of task was already employed in (Weston et al., 2014) in the context of a virtual world. In that sense, the result obtained to such task are comparable with the state of the art approaches.

Yes/No Questions : This task tests the ability of a model to answer true/false type questions like “*Is the food italian ?*”. The conversion of a dialog to such format is deterministic regarding the fact that the utterances and corresponding true states are known at each utterance of a given dialog.

Indefinite Knowledge : This task tests a more complex natural language construction. It tests if statements can be models in order to describe possibilities rather than certainties, as proposed in (Weston et al., 2014). In our case, the answer will be “maybe” to the question “*Is the price-range required moderate ?*” if the slot hasn’t been mentioned yet throughout the current dialog. In the case of state tracking, it will allow to seamlessly deal with unknown information about the dialog state. Concretely, this set of questions and answers are generated has a super-set of the Yes-No Questions set. First, sub-dialog starting from the first utterance of a given dialog are extracted under the condition that a given slot is not informed in the corresponding annotation. Then, a question-answering question is generated.

Counting and Lists/Sets : This last task tests the capacity of the model to perform simple counting operations, by asking about the number of objects with a certain property, e.g. “*How many area are requested ?*”. Similarly, the ability to produce a set of single word answers in the form of a list, e.g. “*What are the area requested ?*” is investigated. Table 1 give an example of each of the question type presented below on a dialog sample of *DSTC-2* corpus.

Inference procedure: Concretely, the current set of utterances of a dialog will be placed into the memory using sentence based encoding and the

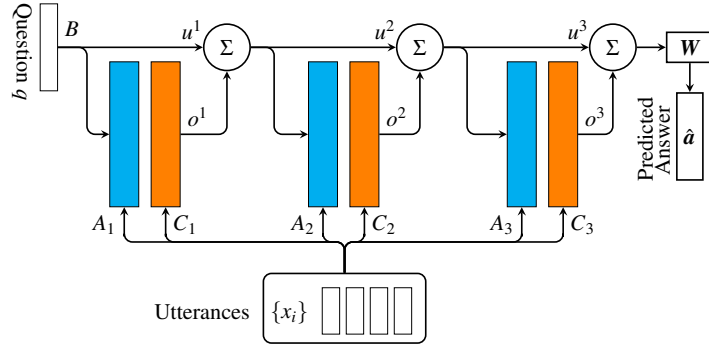


Figure 2: Illustration of the proposed MemN2N based state dialog tracker model with 3 hops.

question will be encoded as the controller state at $t = 1$. The answer will be produced using a softmax operation over the answer vocabulary that is supposed fixed. We consider this hypothesis valid in the case of factoid and list questions because the set of value for a given variable is often considered known. In the cases of Yes/No and Indefinite knowledge question, {Yes, No, Maybe} are added to the output vocabulary. Following (Weston et al., 2014), a list-task answer will be considered as a single element in the answer set and the count question. A possible alternative would be to change the activation function used at the output of the MemN2N from softmax activation function to a logistic one and to use a categorical cross entropy loss. A drawback of such alternative would be the necessity of cross-validating a decision threshold in order to select a eligible answers. Concerning the individual numbers for the count question set, the numbers founded on the training set are added into the vocabulary.

We believe more reasoning capabilities need to be explore in the future, like spacial and temporal reasoning or deduction as suggested in (Weston et al., 2015). However, it will probably need the development of a new dedicated resource. Another alternative could be to develop a question-answering annotation task based on a dialog corpus where reasoning task are present. The closest work to our proposal that can be cited is (Bordes and Weston, 2016). In this paper, the authors defines a so-called End-to-End learnable dialog system to infer an answer from a finite set of eligible answers w.r.t the current list of utterances of the dialog. The authors generate 5 artificial tasks of dialog. However the reasoning capabilities are not explicitly addressed and the author explicitly claim that the resulting dialog system is not satisfactory

yet. Indeed, we believe that having a proper dialog state tracker where a policy is built on top can guarantee dialog achievement by properly optimizing a reward function throughout a explicitly learnt dialog policy. In the case of proper end-to-end systems, the objective function is still not explicitly defined (Serban et al., 2015) and the resulting systems tend to be used in the context of chat-oriented and non-goal oriented dialog systems. In the next section, we present experimental details and results obtained on the basis of the *DSTC-2* dataset and its conversion to the four mentioned reasoning tasks.

4 Experiments

4.1 Dataset and Data Preprocessing

In the *DSTC-2* dialog corpus, a user queries a database of local restaurants by interacting with a dialog system. A dialog proceeds as follows: first, the user specifies constraints concerning the restaurant. Then, the system offers the name of a restaurant that satisfies the constraints. Finally, the user accepts the offer and requests additional information about the accepted restaurant. In this context, the dialog state tracker should be able to track several types of information that compose the state like the geographic area, the food type and the price range slots. In order to make comparable experiments, sub-dialogs generated from the first utterance to each utterance of each dialog of the corpus have been generated. The corresponding question-answer pairs have been generated using the annotated state for each of the sub-dialog. In the case of factoid question, this setting allows for fair comparison at the utterance-level state tracking gains with the prior art. The same protocol has been adopted for the generated reasoning task. In that sense, the tracker task consists

Index	Actor	Utterance
1	Cust	Im looking for a cheap restaurant in the west or east part of town.
2	Agent	Thanh Binh is a nice restaurant in the west of town in the cheap price range.
3	Cust	What is the address and post code.
4	Agent	Thanh Binh is on magdalene street city centre.
5	Cust	Thank you goodbye.
6		Factoid Question What is the pricerange ? Answer: {Cheap}
7		Yes/No Question Is the Pricerange Expensive ? Answer: {No}
8		Indefinite Knowledge Is the FoodType chinese ? Answer: {Maybe}
8		Listing task What are the areas ? Answer: {West,East}

Table 1: : Dialog state tracking question-answering examples from *DSTC2* dataset

in finding the value l^* as defined in Section 3.3. In the overall dialog corpus, Area slot counts 5 possible values, Food slot counts 91 possible values and Pricerange slot counts 3 possible values. In order to exhibit reasoning capability of the proposed model in the context of dialog state tracking, three other dataset have been automatically generated from the dialog corpus in order to support 3 capabilities of reasoning described in Section 3.3. Dialog modification has been required for two reasoning tasks, *List* and *Count*. Two types of rules have been developed to automatically produce modified dialogs. On a first hand, string matching has been performed to determine the position of a slot values in a given utterance and an alternative statement has been produced as a substitution. For example, the utterance “*I’m looking for a chinese restaurant in the north*” can be replaced by “*I’m looking for a chinese restaurant in the north or the west of town*”. A second type of modification has been performed in an inter-utterance fashion. For example, assuming a given value “north” has been informed in the current state of a given dialog, one can add lately in the dialog a remark like “I would also accept a place east side of town”. This kind of statement tends to not affect the overall flow of the dialog and allows to add richer semantic to the dialog. In the future, we plan to develop a richer set of generation procedures to augment the dataset. Nevertheless, we believe this simple dialog augmentation strategy allows to exhibit the competency of the proposed model beyond factoid questions.

4.2 Training Details

As suggested in (Sukhbaatar et al., 2015), 10% of the set was held-out to form a validation set for hyperparameter tuning. Concerning the utterance encoding, we use the so-called *Temporal Encoding* technique. In fact, reading tasks require some notion of temporal context. To enable the model

to address them, the memory vector is modified as such $m_i = \sum_j Ax_{ij} + T_A(i)$, where $T_A(i)$ is the i^{th} row of a dedicated matrix T_A that encodes temporal information. The output embedding is augmented in the same way with a matrix T_c (e.g. $c_i = \sum_j Cx_{ij} + T_C(i)$). Both T_A and T_C are learned during training in an end-to-end fashion. They are also subject to the same sharing constraints as A and C . The embedding matrix A and B are initialized using GoogleNews word2vec embedding model (Mikolov et al., 2013). Also suggested on (Sukhbaatar et al., 2015), utterances are indexed in reverse order, reflecting their relative distance from the question so that x_1 is the last sentence of the dialog. Furthermore, adjacent weight tying schema has been adopted. Learning rate η is initially assigned a value of 0.005 with exponential decay applied every 25 epochs by $\eta/2$ until 100 epochs are reached. Then, *linear start* is used in all our experiments as proposed by (Sukhbaatar et al., 2015). More precisely, the softmax function in each memory layer is removed and re-inserted after 20 epochs. Batch size is set to 16 and gradients with an L_2 norm larger than 40 are divided by a scalar to have norm 40. All weights are initialized randomly from a Gaussian distribution with zero mean and $\sigma = 0.1$. In all our experiments, we have tested a set of the embedding size $d \in \{20, 40, 60\}$. After validation, each model uses a 5-hops depth configuration.

4.3 Experimental results

Table 3 presents tracking accuracy obtained for three variables of the *DSTC2* dataset formulated as *Factoid Question* task. We compare with two established utterance-level discriminative neural trackers, a Recurrent Neural Network (RNN) model (Henderson et al., 2014a) and the Neural Belief Tracker (Mrksic et al., 2016). As suggested in this last work, the first RNN baseline model uses no semantic (i.e. synonym) dictionary, while

Locutor	Utterance	Hop 1	Hop 2	Hop 3	Hop 4	Hop 5
Cust	Im looking for a cheap restaurant that serves chinese food	0.00	0.18	0.11	0.04	0.00
Agent	What part of town do you have in mind	0.33	0.30	0.00	0.00	0.00
Cust	I dont care	0.00	0.00	0.17	0.37	1.00
Agent	Rice house serves chinese food in the cheap price range	0.01	0.00	0.00	0.00	0.00
Cust	What is the address and telephone number	0.58	0.09	0.01	0.00	0.00
Agent	Sure rice house is on mill road city centre	0.03	0.00	0.00	0.00	0.00
Cust	Phone number	0.00	0.00	0.00	0.00	0.00
Agent	The phone number of rice house is 765-239-09	0.02	0.01	0.00	0.00	0.00
Cust	Thank you good bye	0.02	0.42	0.71	0.59	0.00
What is the area ? Answer: dontcare						

Table 2: Attention shifting example for the *Area* slot from *DSTC2* dataset, the values corresponds the p_i values affected to each memory block m_i at each hop of the MemN2N

the improved baseline uses a hand-crafted semantic dictionary designed for the *DSTC2* ontology. In this context, a MemN2N model allows to obtain competitive results with the most close, non-memory enhanced, state of the art approach of recurrent neural network with word embedding as prior knowledge.

Model	Area	Food	Price	Joint
RNN - no dict.	0.92	0.86	0.86	0.69
RNN + sem. dict.	0.91	0.86	0.93	0.73
NBT-DNN	0.90	0.84	0.94	0.72
NBT-CNN	0.90	0.83	0.93	0.72
MemN2N($d = 40$)	0.89	0.88	0.95	0.74

Table 3: **One supporting fact task** : Acc. obtained on DSTC2 test set

As a second result, Table 4 presents the performance obtained for the four reasoning tasks. The obtained results lead us to think that MemN2N are a competitive alternative for the task dialog state tracking but also increase the spectrum of definition of the general dialog state tracking task to machine reading and reasoning. In the future, we believe new reasoning capabilities like spacial and temporal reasoning and deduction should be exploited on the basis of a specifically designed dataset.

5 Conclusion and Further Work

This paper describes a novel method of dialog state tracking based on the paradigm of machine reading and solved using MemN2N, a memory-enhanced neural network architecture. In this context, a dataset format inspired from the current datasets of machine reading tasks has been developed for this task. It is the first attempt to solve this classic sub-problem of dialog management in

Variable	d	Yes-No	I.K.	Count.	List.
Food	20	0.85	0.79	0.89	0.41
	40	0.83	0.84	0.88	0.42
	60	0.82	0.82	0.90	0.39
Area	20	0.86	0.83	0.94	0.79
	40	0.90	0.89	0.96	0.75
	60	0.88	0.90	0.95	0.78
PriceRange	20	0.93	0.86	0.93	0.83
	40	0.92	0.85	0.90	0.80
	60	0.91	0.85	0.91	0.81

Table 4: **Reasoning tasks** : Acc. on DSTC2 reasoning datasets

such way. Beyond the experimental results presented in the experimental section, the proposed approach offers several advantages compared to state of the art methods of tracking. First, the proposed method allows to perform tracking on the basis of segment-dialog-level annotation instead of utterance-level one that is commonly admitted in academic datasets but tedious to produce in a large scale industrial environment. Second, we propose to develop dialog corpus requiring reasoning capabilities to exhibit the potential of the proposed model. In future work, we plan to address more complex tasks like spatial and temporal reasoning, sorting or deduction and experiment with other memory enhanced inference models. Indeed, we plan to experiment and compare the same approach with Stacked-Augmented Recurrent Neural Network (Joulin and Mikolov, 2015) and Neural Turing Machine (Graves et al., 2014) that sounds also promising for these family of reasoning tasks.

References

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad

- Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL.
- Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*.
- Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. Towards an ISO standard for dialogue act annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), may.
- Oren Etzioni, Michele Banko, and Michael J. Cafarella. 2007. Machine reading. In *AAAI Spring Symposium: Machine Reading*. AAAI.
- Milica Gasic and Steve Young. 2011. Effective handling of dialogue state in the hidden information state POMDP-based dialogue manager. *TSLP*, 7(3):4.
- Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden Markov models. *Machine Learning*, 29(2-3):245–273.
- Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing machines. *CoRR*.
- Matthew Henderson, Blaise Thomson, and Steve Young. 2013. *Proceedings of the SIGDIAL 2013 Conference*, chapter Deep Neural Network Approach for the Dialog State Tracking Challenge, pages 467–471. Association for Computational Linguistics.
- M. Henderson, B. Thomson, and S. J. Young. 2014a. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE Spoken Language Technology*.
- Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014b. The third dialog state tracking challenge. In *SLT*, pages 324–329. IEEE.
- Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of SIGDial*.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*.
- Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 190–198.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*.
- Angeliki Metallinou, Dan Bohus, and Jason Williams. 2013. Discriminative state tracking for spoken dialog systems. In *Association for Computer Linguistics*, pages 466–475. The Association for Computer Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Leon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.
- Diego H. Milone and Antonio J. Rubio. 2003. Prosodic and accentual information for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 11(4):321–333.
- Nikola Mrksic, Diarmuid O Seaghda, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *CoRR*, abs/1606.03777.
- Hoifung Poon and Pedro M. Domingos. 2010. Machine reading: A “killer app” for statistical relational AI. In *Statistical Relational Artificial Intelligence*, volume WS-10-06 of *AAAI Workshops*. AAAI.
- Antoine Raux and Yi Ma. 2011. Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems. In *INTERSPEECH*, pages 801–804. ISCA.
- Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, pages 193–203. ACL.
- Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *CoRR*.
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In Corinna Cortes, Neil D. Lawrence,

- Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.
- Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, November.
- Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *CoRR*.
- Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*.
- Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *CoRR*.
- Jason D. Williams, Pascal Poupart, and Steve Young. 2005. Factored partially observable markov decision processes for dialogue management. In *In 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems*, pages 76–82.
- Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *D&D*, 7(3):4–33.
- Jason D. Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *Proceedings of SIGDIAL*. ACL Association for Computational Linguistics, June.
- Peter Z. Yeh, Benjamin Douglas, William Jarrold, Adwait Ratnaparkhi, Deepak Ramachandran, Peter F. Patel-Schneider, Stephen Laverty, Nirvana Tikku, Sean Brown, and Jeremy Mendel. 2014. A speech-driven second screen application for TV program discovery. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 3010–3016. AAAI Press.

Sentence Segmentation in Narrative Transcripts from Neuropsychological Tests using Recurrent Convolutional Neural Networks

Marcos Vinícius Treviso Christopher Shulby Sandra Maria Aluísio
marcostreviso@usp.br cshulby@icmc.usp.br sandra@icmc.usp.br
Interinstitutional Center for Computational Linguistics (NILC)
Institute of Mathematical and Computer Sciences
University of São Paulo

Abstract

Automated discourse analysis tools based on Natural Language Processing (NLP) aiming at the diagnosis of language-impairing dementias generally extract several textual metrics of narrative transcripts. However, the absence of sentence boundary segmentation in the transcripts prevents the direct application of NLP methods which rely on these marks to function properly, such as taggers and parsers. We present the first steps taken towards automatic neuropsychological evaluation based on narrative discourse analysis, presenting a new automatic sentence segmentation method for impaired speech. Our model uses recurrent convolutional neural networks with prosodic, Part of Speech (PoS) features, and word embeddings. It was evaluated intrinsically on impaired, spontaneous speech, as well as, normal, prepared speech, and presents better results for healthy elderly (CTL) ($F_1 = 0.74$) and Mild Cognitive Impairment (MCI) patients ($F_1 = 0.70$) than the Conditional Random Fields method ($F_1 = 0.55$ and 0.53 , respectively) used in the same context of our study. The results suggest that our model is robust for impaired speech and can be used in automated discourse analysis tools to differentiate narratives produced by MCI and CTL.

1 Introduction

Mild Cognitive Impairment (MCI) has recently received much attention, as it may represent a pre-clinical state of Alzheimer’s disease (AD). MCI can affect one or multiple cognitive domains (e.g. memory, language, visuospatial skills and the ex-

ecutive function); the kind that affects memory, called amnesic MCI, is the most frequent and that which most often converts to AD (Janoutová et al., 2015). As dementias are chronic progressive diseases, it is important to identify them in the early stages, because early detection yields a greater chance of success for non-pharmacological treatment strategies such as cognitive training, physical activity and socialization (Teixeira et al., 2012). The definition of MCI diagnostic criteria is conducted mainly by the cognitive symptoms presented by patients in standardized tests and by functional impairments in daily life (McKhann et al., 2011). Difficulties related with narrative discourse deficits (e.g. repetitions or gaps during the narrative) may lead an elderly individual to look for a specialist. Narrative discourse is the reproduction of an experienced episode (necessarily evoking memory), respecting temporal and causal relations among events. Although MCI is clinically characterized by episodic memory deficits, language impairment may also occur.

Certain widely used neuropsychological tests require patients to retell or understand a story. This is the case of the logical memory test, where one reproduces a story after listening to it. The higher the number of recalled elements from the narrative, the higher the memory score (Wechsler, 1997; Bayles and Tomoeda, 1991; Morris et al., 2006). However, the main difficulties in applying these tests are: (i) time required, since it is a manual task; and (ii) the subjectivity of the clinician. Therefore, automatic analysis of discourse production is seen as a promising solution for MCI diagnosis, because its early detection ensures a greater chance of success in addressing potentially reversible factors (Muangpaisan et al., 2012). Since discourse is a natural form of communication, it favors the observation of the patient’s functionality in everyday life. Moreover, it

provides data for observing the language-cognitive skills interface, such as executive functions (planning, organizing, updating and monitoring data).

With regard to the Wechsler Logical Memory (WLM) test, the original narrative used is short, allowing for the use of Automatic Speech Recognition (ASR) output even without capitalization and sentence segmentation, as shown by Lehr et al. (2012) for English. They based their method on automatic alignment of the original and patient transcripts in order to calculate the number of recalled elements.

The evaluation of narrative discourse production from the standpoint of linguistic impairment is an attractive alternative as it allows for linguistic microstructure analysis, including phonetic-phonological, morphosyntactic and semantic-lexical components, as well as semantic-pragmatic macrostructures. Automated discourse analysis tools based on Natural Language Processing (NLP) resources and tools aiming at the diagnosis of language-impairing dementias via machine learning methods are already available for the English language (Fraser et al., 2015b; Yancheva et al., 2015; Roark et al., 2011) and also for Brazilian Portuguese (BP) (Aluísio et al., 2016). The latter study used a publicly available tool, Coh-Matrix-Dementia¹, to extract 73 textual metrics of narrative transcripts, comprising several levels of linguistic analysis from word counts to semantics and discourse. However, the absence of sentence boundary segmentation in transcripts prevents the direct application of NLP methods that rely on these marks in order for the tools to function properly. To our knowledge, only one study evaluating automatic sentence segmentation in English transcripts of elderly aphasic exists (Fraser et al., 2015a).

The purpose of this paper is to present our method, DeepBond, for automatic sentence segmentation of spontaneous speech of healthy elderly (CTL) and MCI patients. Although it was evaluated for BP data, it can be adapted to other languages as well.

2 Related Work

The sentence boundary detection task has been treated by many researchers. Liu et al. (2006) investigated the imbalanced data problem, since there are more non-boundary words than not; their

study was carried out using two speech corpora: conversational telephone and broadcast news, both for English.

More recent studies have focused on Conditional Random Field (CRF) and Neural Network models. Wang et al. (2012) and Hasan et al. (2014) use CRF based methods to identify word boundaries in speech corpora datasets, more specifically on English broadcast news data and English conversational speech (lecture recordings), respectively. Khomitsevich et al. (2015), similar to our work, used a combination of two models, one based on Support Vector Machines to deal with prosodic information, and other based on CRF to deal with lexical information. They combine the two models using a logistic regression classifier.

Xu et al. (2014) uses a combination of CRF and a Deep neural network (DNN) to detect sentence boundaries on broadcast news data. Che et al. (2016) uses two different convolutional neural network (CNN), one which moves in only one dimension and another which moves in two. They achieved good results on a TED talks dataset. Tilk and Alumäe (2015) use a recurrent neural network (RNN) with long short-term memory units to restore punctuation in speech transcripts from broadcast news and conversations.

Although there are proposed methods for sentence segmentation of Portuguese datasets (Silla Jr. and Kaestner, 2004; Batista and Mamede, 2011; López and Pardo, 2015), none of them are used for transcriptions produced in a clinical setting for the elderly with dementia and related syndromes. The study most similar to our scenario is (Fraser et al., 2015a), which proposes a segmentation method for aphasic speech based on lexical, PoS and prosodic features using tools and a generic acoustic model trained for English. Their approach is based on a CRF model, and the best results for this study were obtained for non-spontaneous broadcast news data.

Our method uses recurrent convolutional neural networks with prosodic, PoS features, and also word embeddings and was evaluated intrinsically on impaired, spontaneous speech and normal, prepared speech. Although DNNs have already been used for this task, our work was the first, to the best of our knowledge, to evaluate them on impaired speech.

¹<http://143.107.183.175:22380/>

3 Datasets

A total of 60 participants from a research project on diagnostic tools for language impaired dementias produced narratives used to evaluate our method. Two datasets were used to train our model (Sections 3.1 and 3.2). As a preprocessing step we have removed capitalization information and in order to simulate high-quality ASR, we left all speech disfluences intact. Demographic information for participants in our study is presented in Table 1. A third dataset was used in robustness tests (Section 3.3).

Info	CTL	MCI	AD
Avg. Age	74.8	73.3	78.2
Avg. Education	11.4	10.8	8.6
No. of Male/Female	4/16	6/14	10/10

Table 1: Demographic information of participants in the Cinderella dataset. The Avg. Education is given in years.

3.1 The Cinderella Dataset

The Cinderella dataset consists of spontaneous speech narratives produced during a test to elicit narrative discourse with visual stimuli, using a book consisting of sequenced pictures based on the Cinderella story. In the test, an individual verbally tells the story to the examiner based on the pictures. The narrative is manually transcribed by a trained annotator who scores the narrative by counting the number of recalled propositions.

This dataset consists of 60 narrative texts from BP speakers, 20 controls, 20 with AD, and 20 with MCI, diagnosed at the Medical School of University of São Paulo and also used in Aluísio et al. (2016). Counting all patient groups, this dataset has an audio duration of 4h and 11m, an average of $1843/60 = 30.72$ sentences per narrative, and sentence averages of $23807/1843 = 12.92$ words. AD narratives were only used for training the lexical model.

3.2 The Brazilian Constitution Dataset

This dataset was made available by the LaPS (Signal Processing Laboratory) at the Federal University of Pará (Batista, 2013), and is composed of articles from Brazil’s 1988 constitution, in which the speech is prepared and read. Each file has an averages 30 seconds.

A preprocessing step removed lexical tips which indicate the beginning of the articles, sections and paragraphs. This removal was carried out on both the transcripts and audio. In addition, we separated the new dataset organized by articles, totaling 357 texts. Then, we marked the end of each article and paragraph and inserted punctuation at the end. Titles and chapters have been ignored during this process. We randomly selected 60 texts from this dataset, forcing only the condition that the number of sentences of each text sentence was greater than 12. We refer to the large dataset as Constitution L, and the dataset with the 60 texts as Constitution S.

The average number of sentences in each text of Constitution L is $2698/357 = 7.56$, and the average size of these sentences have $63275/2698 = 23.45$ words while Constitution S has on average $1409/60 = 23.48$ sentences, and these sentences average $30521/1409 = 21.66$ words. The total audio duration of Constitution L is 7h 39m, and Constitution S is 3h 43m.

3.3 The Dog Story Dataset

The Dog Story dataset is available from the BALE (Battery of Language Assessment in Aging, in English) instrument, described in (Jerônimo, 2016). It is composed of transcriptions from the narrative production test based on the presentation of a set of seven pictures telling a story of a boy who hides a dog that he found on the street (Le Boeuf, 1976). This battery was chosen because its aim is to allow for its administration to elderly people who are illiterate and/or of low educational level, who represent the majority of the aged sample assisted by the public health system in Brazil.

This dataset consists of 10 narratives transcripts (6 CTL and 4 MCI), where the average number of sentences and the average size of the sentences are 16.60 and 6.58, respectively. When compared with the Cinderella dataset, the dataset is composed of less sentences and the sentences have fewer words on average.

4 Features

4.1 Lexical features

We divide our lexical features into two groups: PoS features and word embeddings, where every word is represented in a high dimensionality continuous vector.

The PoS features were extracted using a BP

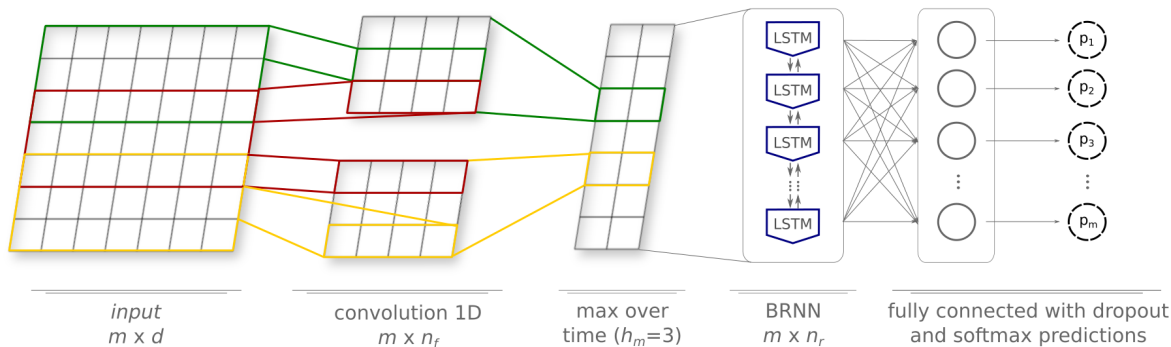


Figure 1: Architecture of the RCNN for both lexical and prosodic model.

morphosyntactic tagger called `nlpnet`² trained on a revised version of the Mac-Morpho corpus (Fonseca et al., 2015), which contains a set of 25 tags.

The word embeddings used in this work have 50 dimensions and were trained by Fonseca et al. (2015) with articles from the BP version of Wikipedia and a large journalistic corpus with articles from the news site G1³, totaling 240 million tokens and a vocabulary of 160,270 words. All of these tokens were made lowercase and trained with a neural language model described in (Collobert et al., 2011).

4.2 Prosodic features

We used three prosodic features: F0, intensity and duration which were extracted at the phonetic level using PRAAT (Boersma and others, 2002) from forced alignment output. Alignment was done using the HTK toolkit (Young et al., 2002) with clean speech corpora and a pronunciation dictionary phonetically transcribed by Petrus (Serani, 2015) and augmented by our rule-based algorithm to insert multiple pronunciations, rendering a suitable model for ASR. The features were calculated for the first, last, penultimate and antepenultimate vowels of each word and pauses. These vowels were chosen based on knowledge of the BP which typically exhibits stress on the penultimate vowel, with notable patterns observed for final vowel stressing, for example words ending in “i” (“Barueri”) or a nasal consonant (“Renan”), and the antepenultimate vowel (usually indicated by a stress diacritic) like “helicóptero” (“helicopter”), “espírito” (“spirit”) and “árvore” (“tree”). Also, Portuguese, like most western languages, distinguishes sentence types by rising and falling pitch patterns, giving the listener a clue as

to whether the speaker has finished a sentence or not. Pause duration was also calculated since the length of a pause can be indicative of the presence of a punctuation mark (Beckman and Ayers Elam, 1997).

5 Model description

To automatically extract features from the input and also deal with the problem of long dependencies between words, we propose a model based on recurrent convolutional neural networks (RCNN), which was inspired by the work of Lai et al. (2015). The architecture of our model can be seen in Figure 1. First, we show how to prepare the input for the network, then we go through the networks layers and describe the training procedure, finally, we discuss the experimental settings.

5.1 Input preparation

In our approach, the input to the network is a transcribed narrative which is categorized as CTL (healthy elderly individuals) and MCI (MCI patients). The narratives contain a sequence of words w_1, w_2, \dots, w_m . Each word is annotated with a label, to indicate whether it precedes a boundary ($y = B$) or not ($y = NB$). We do not make a distinction between punctuation marks, so a boundary is defined as a period, exclamation mark, question mark, colon or semicolon. With this approach, we can see this task as a binary classification problem.

5.2 Representation

Our input contains transcribed narratives with m words in it. We represent the narrative i as $X_i \in \mathbb{R}^{m \times n}$, $X_i = \{x_1, x_2, \dots, x_{m \times n}\}$, where n is the number of features. We represent the boundaries as $Y_i \in \mathbb{R}^2$, $Y_i = \{0, 1\}$, where 0 stands for NB and 1 denotes B . Our final model consists of a

²nilc.icmc.usp.br/nlpnet/

³g1.globo.com/

combination of two models. The first model is responsible for treating only lexical information, while the second treats only prosodic information. Both models have the same architecture shown in Figure 1. This strategy is based on the idea that we can train the lexical model with even more data, since textual information is easily found on the web. In order to obtain the most probable class y for the w_j word, a linear combination was created between these two models, where one receives the weighted complement of the other:

$$\alpha \cdot P_{lexical}(y | w_j) + (1 - \alpha) \cdot P_{prosodic}(y | w_j) \quad (1)$$

Then, the most probable class is the one that maximizes the linear combination from previous equation.

5.2.1 Embedding layer

The data input for the lexical model is divided into two features: word embeddings with dimensions $|e_w|$, and the PoS tags with dimensions $|e_t|$. Given a word w , the respective embedding $e_w \in E_{word}$ is fetched and concatenated with the word's PoS vector $e_t \in E_{tag}$, thus obtaining a new vector size $d = |e_w| + |e_t|$. Out of vocabulary words share a single and randomly generated vector that represents an unknown word.

In the prosodic model we directly feed information about pitch, intensity and duration from the first, last, penultimate and ante-penultimate vowels of each word. Moreover, we feed the information about pause duration after each word, where duration of zero seconds denotes no pause. Therefore, for the prosodic model, we have a vector with dimensions $d = 4 \cdot 3 + 1 = 13$.

5.2.2 Convolutional and pooling layer

Once we have a matrix formed by the features of the words in the text, the convolutional layer receives it, which, in turn, is responsible for the automatic extraction of n_f new features depending on h_c neighboring words (Kim, 2014). The convolutional layer produces a new feature c_j by applying a filter $W \in \mathbb{R}^{h_c \cdot d}$ to a window of h_c words $x_{j-h_c+1:j}$ in a sentence with length m :

$$c_j = f(Wx_{(j-h_c+1):j} + b), \quad h_c \leq j \leq m \quad (2)$$

Where $b \in \mathbb{R}$ represents a bias term and f is a non-linear function.

Our convolutional layer simply moves one dimension vertically, making one step at a time,

which gives us $m - h_c + 1$ generated features. Since we want to classify exactly m elements, we add $p = \lfloor h_c/2 \rfloor$ zero-padding on both sides of the text. Applying this strategy for each entry x_j yields the complete feature map $c \in \mathbb{R}^{(m-h_c+1)+2 \cdot p}$.

In addition, we apply a max-pooling operation over time, looking at a region of h_m elements to find the most significant features:

$$\hat{c} = \max_{1 \leq j \leq m} \{c_{(j-h_m+1):j}\} \quad (3)$$

5.2.3 Recurrent layer

The new features extracted are fed into a recurrent bidirectional layer which has n_r units. A recurrent layer is able to store historic information by connecting the previous hidden state with the current hidden state at a time t . The values in the hidden and output layers are computed as follows:

$$h_t = f(W_x x_t + W_h h_{t-1} + b_h) \quad (4)$$

$$y_t = g(W_y h_t + b_y) \quad (5)$$

where W_x , W_h , and W_y are the connection weights, b_y and b_h are bias vectors, and f and g are non-linear functions. Here, we use a special unit known as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), which is able to learn over long dependencies between words by a purpose-built memory cell. Figure 2 shows a single LSTM memory cell.

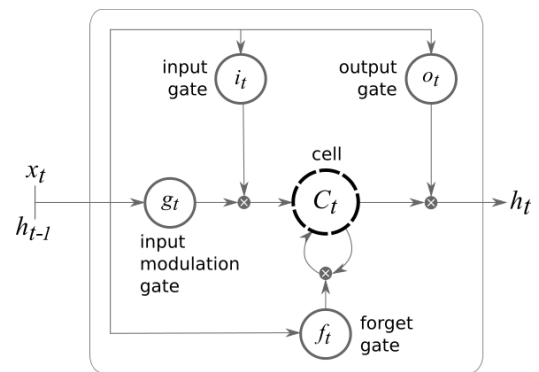


Figure 2: Diagram of a LSTM memory cell.

The LSTM updates for time steps t are done as described by Jozefowicz et al. (2015), which is a slight simplification of the one described by Graves and Jaitly (2014), where the memory cell is implemented as follows:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
g_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}$$

where $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function, $h_t \in \mathbb{R}^{n_r}$ is the hidden unit, $i_t \in \mathbb{R}^{n_r}$ is the input gate, $f_t \in \mathbb{R}^{n_r}$ is the forget gate, $o_t \in \mathbb{R}^{n_r}$ is the output gate, $g_t \in \mathbb{R}^{n_r}$ is the input modulation gate, and $c_t \in \mathbb{R}^{n_r}$ is the memory cell unit, which is the summation of the previous memory cell modulated by the forget gate f_t , and a function of the current input with previous hidden state modulated by the input gate i_t .

As in Graves and Jaitly (2014), we used the features by looking at forward states and backward states. This kind of mechanism is known as a bidirectional neural network (BRNN), since it learns weights based on both past and future elements given a timestep t . In order to implement the BRNN, we reversed the sentences as a trick before we fed them to a regular LSTM layer, doubling the number of weights used in the recurrent layer. The output from this layer is the summation of the forward output with backward output:

$$y_t = \overleftarrow{y}_t + \overrightarrow{y}_t \quad (6)$$

With a bidirectional LSTM layer, we are able to explore the principle that words nearby have a greater influence in classification, while considering that words farther away can also have some impact. This often happens, for example, in the case of question words and conjunctions: por que (“why”); qual (“which”); quem (“who”); quando (“when”), etc.

5.2.4 Fully connected layer

After the BRNN layer, dropout is used to prevent co-adaptation of hidden units during forward-backpropagation, where we ignore some neurons meaning to reduce the chance of overfitting the model (Srivastava et al., 2014).

The last layer receives the output from the BRNN in each timestep and passes them through a fully connected layer, where the softmax operation is calculated, giving us the probability of whether

or not the word precedes a boundary:

$$\hat{y}_t = \text{softmax}(W y_t + b) \quad (7)$$

Where $W \in \mathbb{R}^{n_r \times 2}$ is a matrix of weights, $b \in \mathbb{R}^{n_r}$ is a bias vector, and softmax is defined as:

$$s_j(z) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, \quad \text{for } j = 1, 2, \dots, K \quad (8)$$

5.3 Training

We define all of the parameters to be trained as θ .

$$\begin{aligned}
\theta = \{ & E_{word}, E_{tag}, W^{(c)}, b^{(c)}, W^{(f)}, \\
& b^{(f)}, \overleftarrow{W}^{(r)}, \overleftarrow{b}^{(r)}, \overrightarrow{W}^{(r)}, \overrightarrow{b}^{(r)} \} \quad (9)
\end{aligned}$$

Where $E_{word} \in \mathbb{R}^{|V| \times |e_w|}$ is the lookup table for the word embeddings, $E_{tag} \in \mathbb{R}^{|V_{tag}| \times |e_t|}$ is the lookup table for PoS tags, and $|V|, |V_{tag}|$ represents the size of the vocabulary for word embeddings and PoS tags, respectively.

For the convolutional layer: the weights $W^{(c)} \in \mathbb{R}^{n_f \times h_c \times d}$ and the bias vector $b^{(c)} \in \mathbb{R}^{n_f}$.

For the fully connected layer: the weights matrix $W^{(f)} \in \mathbb{R}^{n_r \times 2}$ and the bias vector $b^{(f)} \in \mathbb{R}^{n_r}$.

For the BRNN layer we divide the set of parameters from BRNN into two sets. Those from the forward pass and backward pass. Each set contains the weights for an input $W_x^{(r)} \in \mathbb{R}^{n_r \times n_f}$, the weights for previous hidden states $W_h^{(r)} \in \mathbb{R}^{n_r \times n_r}$, and the bias vectors $b^{(r)} \in \mathbb{R}^{n_r}$ for all gates (i, f, o, g). Additionally, we have the weights for an output in a timestep $W_y^{(r)} \in \mathbb{R}^{n_r \times n_r}$ and a bias vector $b_y \in \mathbb{R}^{n_r}$.

We define the loss function \mathcal{L} as categorical cross-entropy (Murphy, 2012), shown in the equation below, which aims to minimize the negative log likelihood in relation to the weights. Since we have an unbalanced class problem, we give different weights for each class, where the weight of the minority class (B) is greater than that of the majority (NB).

$$\mathcal{L}(y, \hat{y}) = - \sum_i y_i \log(\hat{y}_i) c w_{y_i} \quad (10)$$

Where y are our real targets, \hat{y} are our predictions, and cw are the class weights for $\ell = B$ and $\ell = NB$, calculated as follows:

$$c w_\ell = \frac{|y|}{2 \cdot |y = \ell|} \quad (11)$$

We minimize the loss function with respect to all weights $\theta \mapsto \mathcal{L}$ by using RMSProp algorithm

(Tieleman and Hinton, 2012) with backpropagation to compute the gradients $\nabla \mathcal{L}$. The update step for a timestep t is made by normalizing the gradients by an exponent moving at an average r_t :

$$r_t = \gamma r_{t-1} + (1 - \gamma) \nabla \mathcal{L}(\theta_t)^2 \quad (12)$$

$$\theta_{t+1} = \theta_t - \eta \frac{\nabla \mathcal{L}(\theta_t)}{\sqrt{r_t} + \epsilon} \quad (13)$$

Where η is the learning rate and $0 < \gamma < 1$ is the forgetting factor.

5.4 Experiment settings

We break the text in tokens delimited by spaces. We do not remove stopwords from the texts, since they can be important features for our domain.

We ran a 5-fold cross-validation for the group being analyzed (CLT or MCI), which leaves about 10% of the data for testing, the rest for training.

The weight matrix for tag embeddings E_{tag} was generated randomly from a gaussian distribution scaled by fan in + fan out (Glorot and Bengio, 2010). Both embeddings matrix E_{word} and E_{tag} were adjusted during training. We follow previous studies on sentence boundary detection to set the network hyper-parameters (Tilk and Alumäe, 2015; Che et al., 2016). The values for each parameter are shown in Table 2.

Var.	Parameter	Lexical	Prosodic
$ e_w $	Word emb. size	50	-
$ e_t $	Tag emb. size	10	-
n_f	Conv. filters	100	8
h_c	Filter length	7	5
h_m	Max-pool size	3	3
n_r	Recurrent units	100	100
γ	Forget factor	0.9	0.9
η	Learning rate	0.001	0.001

Table 2: RCNN Hyper-parameters.

We tried three different learning rate values $\eta \in \{0.01, 0.003, 0.001\}$ for both lexical and prosodic models, and found that 0.001 yielded best results. We trained our network over 20 epochs using a bucket strategy, which groups training examples in buckets of similar sentence size. Our implementation is based on Theano (Bergstra et al., 2010), a library that defines, optimizes and evaluates mathematical expressions in an effective way.

6 Evaluation

We evaluated our method intrinsically and also compared it with the method developed by Fraser et al. (2015a) for all of the datasets. We also performed robustness tests to indicate how well our method responds to both (i) test data that varies from Cinderella training data and (ii) train data that varies from Cinderella testing data.

If we classified all words as NB , our method would have an accuracy superior to 90%. For this reason, we use the F_1 metric, which is defined as the harmonic mean between precision and recall. And since we are more interested in knowing whether our method correctly identifies the boundaries, we ignore the NBs and calculate F_1 only for the positive class (B).

6.1 Results

In this subsection, we evaluate the performance of our classifier (RCNN) for the Cinderella and Constitution datasets. Table 3 summarizes the results.

From Table 3 we can see that our approach presents better results for the Constitution dataset than Cinderella. This may be related to the text quality, as the Cinderella transcripts presents many disfluences, characteristic of spontaneous speech. As expected, results for CTL were higher than for MCI, since CTL narratives contain less disfluencies. Another important observation is that our method performs much better than the baseline. Where the baseline represents the results for a classifier that predicts all words as B . The Constitution results show us that traditional machine learning techniques used in NLP can be applied to this scenario, since the differences in the Cinderella data are few. Another reason that supports this statement is that F_1 results from related studies on sentence boundary detection based on well-written texts are between 0.7 and 0.8 for two classes (Wang et al., 2012; Khomitsevich et al., 2015; Tilk and Alumäe, 2015; Che et al., 2016). When we compare the Constitution size relation we find out that corpus size is not greatly affected by the results, since the results for Constitution S were slightly better than for Constitution L. We think that, even with less data, our method performs better on Constitution S because of the distribution of sentence quantity in the dataset, where Constitution S has an average of 23.48 sentences per text, while Constitution L has an average of only 7.56 sentences per text.

Features	Cinderella						Constitution					
	CTL			MCI			L			S		
	<i>P</i>	<i>R</i>	<i>F</i> ₁	<i>P</i>	<i>R</i>	<i>F</i> ₁	<i>P</i>	<i>R</i>	<i>F</i> ₁	<i>P</i>	<i>R</i>	<i>F</i> ₁
Baseline	0.07	1.00	0.13	0.08	1.00	0.14	0.03	1.00	0.07	0.04	1.00	0.08
PoS	0.36	0.82	0.50	0.32	0.83	0.46	0.30	0.89	0.44	0.29	0.79	0.42
Prosody	0.20	0.59	0.30	0.19	0.58	0.29	0.54	0.84	0.66	0.48	0.85	0.61
Embeddings	0.70	0.70	0.70	0.63	0.77	0.69	0.60	0.63	0.63	0.60	0.64	0.62
PoS + Pros.	0.40	0.74	0.52	0.36	0.80	0.49	0.52	0.91	0.66	0.57	0.85	0.68
Emb. + PoS	0.71	0.72	0.71	0.64	0.75	0.69	0.64	0.72	0.68	0.63	0.67	0.65
Emb. + Pros.	0.71	0.74	0.72	0.64	0.77	0.70	0.71	0.83	0.76	0.74	0.81	0.77
All	0.72	0.76	0.74	0.66	0.74	0.70	0.77	0.82	0.79	0.76	0.85	0.80

Table 3: F_1 for boundary class for each feature set on Cinderella and Constitution data using our method.

We also evaluated the performance of different feature sets with our datasets. Embeddings have a great impact on both datasets. The PoS information was influential on both datasets, but by a small margin, since it has a small difference when used with embeddings (0.01) on the Cinderella, and (0.03) Constitution data. This tells us that embeddings already bring enough morphosyntactic information. It is evident that the weight of the prosodic features is higher on Constitution, which is based on prepared speech, than in Cinderella. This result is consistent with those found by Kolár et al. (2009) and Fraser et al. (2015a). We also believe that the quality of the audio recordings may have impacted the weight of the prosodic features, since the Constitution dataset was recorded by speech processing experts in a studio and the Cinderella dataset was recorded in a clinical setting. In light of this, we can see that our method performs better when all features are used. Furthermore, the best results were obtained by using $\alpha = 0.6$, from the linear combination in Equation 1, showing that our model lends more weight to the lexical model.

6.2 Comparison of methods

In order to compare our model with related work, we replicated the approach proposed by Fraser et al. (2015a), which uses a CRF model for sentence segmentation. To explain the choice for a recurrent convolutional model, we split our method in three: (i) Multilayer Perceptron (MLP): we removed the convolutional and the recurrent layer of our model, and added a hidden fully-connected layer with 100 units and sigmoid activation; (ii) CNN: we simply removed the recurrent layer from our model and passed the output from the convolutional to the fully-connected layer; (iii) Recurrent

Neural Network (RNN): analogous to the CNN model, we removed the convolutional layer and connected the embedding layer with the recurrent layer. The results for each method are presented in Table 4.

Our method achieved the best results in both datasets. We can see that the CRF method, used by Fraser et al. (2015a), obtained the worst results on Constitution, and was only better than RNN on the Cinderella data. These results were similar to those reported in their paper, which suggests that our replication was faithful. We believe that the RNN performed poorly because it has a large set of weights to be trained, and since we have relatively little data, it failed to achieve good results. This may be related to the fact that LSTM units are very complex and need more data to be able to converge. Looking at the Constitution results, which have about three times more words than the Cinderella data, we can note the difference (~ 0.2) with relation to corpus size.

MLP and CNN alone were able to achieve better results than CRF and RNN, but MLP results for the MCI subset were not as good as CNN, which indicates that MLP alone is not able to deal with narratives that are potentially impaired. However, for the Constitution data, MLP obtained results very close (~ 0.02) to our best method.

Our RCNN achieved the best results on both datasets, implying that a union of these models was a good choice in order to deal with impaired speech. We believe that the greatest influence was from the CNN, and the addition of a recurrent layer with LSTM was able to deal with some particular cases, likely over long dependencies similar to the findings in (Tilk and Alumäe, 2015), where the CNN was not able to do so due to the fixed filter length in the convolution process, a re-

Methods	Cinderella						Constitution					
	CTL			MCI			L			S		
	<i>P</i>	<i>R</i>	F_1	<i>P</i>	<i>R</i>	F_1	<i>P</i>	<i>R</i>	F_1	<i>P</i>	<i>R</i>	F_1
CRF	0.70	0.45	0.55	0.62	0.46	0.53	0.89	0.36	0.51	0.84	0.34	0.48
MLP	0.59	0.79	0.67	0.47	0.80	0.59	0.75	0.79	0.77	0.76	0.80	0.78
RNN	0.27	0.68	0.39	0.73	0.25	0.37	0.43	0.92	0.58	0.44	0.85	0.57
CNN	0.64	0.79	0.71	0.59	0.77	0.67	0.65	0.85	0.73	0.58	0.89	0.70
RCNN	0.72	0.76	0.74	0.66	0.74	0.70	0.77	0.82	0.79	0.76	0.85	0.80

Table 4: Best F_1 results for each method.

sult which was also noted in (Che et al., 2016).

6.3 Robustness tests

Robustness was evaluated by measuring F_1 on both out-of-genre and in-genre data. The results for each configuration are presented in Table 5.

Trained on	Tested on	<i>P</i>	<i>R</i>	F_1
Constitution	Cinderella CTL	0.19	0.29	0.23
Constitution	Cinderella MCI	0.20	0.25	0.22
Cinderella	Dog story CTL	0.72	0.62	0.66
Cinderella	Dog story MCI	0.65	0.64	0.64

Table 5: Results for robustness tests

We evaluated our method by changing the corpus genre: training with the Constitution and testing with the Cinderella dataset. This evaluation shows that our method performed poorly in this scenario, probably because the differences in the lexical clues between these datasets are high, since the Constitution is composed of prepared speech and Cinderella of spontaneous speech. When we maintain the corpus genre but change the story used in the neuropsychological test, our method can still achieve good results, yielding a small difference of 0.08 for CTL and 0.06 for MCI from our best results. We believe that these results are related with the linear combination weight from Equation 1, where the results were obtained by using $\alpha = 0.8$, lending less weight to the prosodic model when compared to our best results (where it has 40% of influence). Since the Dog Story and Cinderella datasets are composed of spontaneous speech, the lexical clues found in this kind of speech helped the method to achieve good performance.

7 Conclusions and Future Work

We have shown that our model, using a recurrent convolutional neural network, is benefited by

word embeddings and can achieve promising results even with a small amount of data. We found that our method is better for cases where speech is planned, since the prosodic features lend more weight to the classification. Our method achieved good results on impaired speech transcripts even with little data, with an F_1 result of 0.74 on CTL patients, which is comparable with the results from other studies using broadcast news and conversational data (Wang et al., 2012; Khomitsevich et al., 2015; Tilk and Alumäe, 2015; Che et al., 2016). Moreover, our method achieved good results in robustness tests when we changed the story used in the neuropsychological test.

As for future work, we plan to evaluate our method on English data for comparison with related work. Also, we plan on using more text data to train the lexical model, as it is independent from the prosodic model and lends more weight in our evaluations. Moreover, we will evaluate our method with the output of an ASR system for BP, as a higher word recognition error rate can greatly affect our results. Lastly, we would like to evaluate our method with datasets with higher quality audio, more robust acoustic models and a manually aligned portion of the database as better audio segmentation would greatly improve the model and the usefulness of prosodic features.

With respect to improvements in the corpus, our dataset consists of spontaneous speech narratives and was annotated only with periods. Since there are initial conjunctions such as “and”, “moreover”, and “however”, we could include commas. This would turn our problem into a ternary problem. This could be done by increasing the number of neurons in the last layer of our architecture.

Acknowledgments

We thank CNPq for a scholarship granted to the first author.

References

- S. Aluísio, A. Cunha, and C. Scarton. 2016. Evaluating progression of alzheimer’s disease by regression and classification methods in a narrative language test in portuguese. *International Conference on Computational Processing of the Portuguese Language*, pages 374–384, July.
- Fernando Batista and Nuno Mamede. 2011. *Recovering Capitalization and Punctuation Marks on Speech Transcriptions*. Ph.D. thesis, Instituto Superior Técnico.
- Pedro dos Santos Batista. 2013. Avanços em reconhecimento de fala para português brasileiro e aplicações: ditado no libreoffice e unidade de resposta audível com asterisk.
- Kathryn Bayles and C.K. Tomoeda. 1991. *ABCD: Arizona Battery for Communication Disorders of Dementia*. Tucson, AZ: Canyonlands Publishing.
- Mary E. Beckman and Gayle Ayers Elam. 1997. Guidelines for tobi labelling: The ohio state university research foundation.
- James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.
- Paul Boersma et al. 2002. Praat, a system for doing phonetics by computer. *Glott international*, 5(9/10):341–345.
- Xiaoyin Che, Cheng Wang, Haojin Yang, and Christoph Meinel. 2016. Punctuation prediction for unsegmented transcript based on word vector. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Erick R. Fonseca, João Luís G. Rosa, and Sandra Maria Aluísio. 2015. Evaluating word embeddings and a revised corpus for part-of-speech tagging in portuguese. *Journal of the Brazilian Computer Society*, 21(1):1.
- Kathleen C. Fraser, Naama Ben-David, Graeme Hirst, Naida Graham, and Elizabeth Rochon. 2015a. Sentence segmentation of aphasic speech. In *Proceedings of the NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 862–871.
- Kathleen C. Fraser, Jed A. Meltzer, and Frank Rudzicz. 2015b. Linguistic features identify alzheimer’s disease in narrative speech. *Journal of Alzheimer’s Disease*, 49(2):407–422.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.
- Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772.
- Madina Hasan, Rama Doddipatla, and Thomas Hain. 2014. Multi-pass sentence-end detection of lecture speech. In *INTERSPEECH*, pages 2902–2906.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Jana Janoutová, Omar Serý, Ladislav Hosák, and Vladimír Janout. 2015. Is mild cognitive impairment a precursor of alzheimer’s disease? short review. *Central European journal of public health*, 23(4):365.
- Gislaine Machado Jerônimo. 2016. *Produção de narrativas orais no envelhecimento sadio, no comprometimento cognitivo leve e na doença de Alzheimer e sua relação com construtos cognitivos e escolaridade*. Ph.D. thesis.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*.
- Olga Khomitsevich, Pavel Chistikov, Tatiana Krivosheeva, Natalia Epimakhova, and Irina Chernykh. 2015. Combining prosodic and lexical classifiers for two-pass punctuation detection in a russian asr system. In *International Conference on Speech and Computer*, pages 161–169. Springer.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.
- Jáchym Kolár, Yang Liu, and Elizabeth Shriberg. 2009. Genre effects on automatic sentence segmentation of speech: A comparison of broadcast news and broadcast conversations. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4701–4704. IEEE.
- Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI’15*, pages 2267–2273. AAAI Press.
- Christine Le Boeuf. 1976. *Raconte: 55 historiettes en images*. L’École.

- Maidier Lehr, Emily Tucker Prudhommeaux, Izhak Shafran, and Brian Roark. 2012. Fully automated neuropsychological assessment for detecting mild cognitive impairment. In *INTERSPEECH*, pages 1039–1042.
- Yang Liu, Nitesh V. Chawla, Mary P. Harper, Elizabeth Shriberg, and Andreas Stolcke. 2006. A study in machine learning from imbalanced data for sentence boundary detection in speech. *Computer Speech and Language*, 20(4):468–494.
- Roque López and Thiago A.S. Pardo. 2015. Experiments on sentence boundary detection in user-generated web content. In *Computational Linguistics and Intelligent Text Processing*, pages 227–237.
- Guy M. McKhann, David S. Knopman, Howard Chertkow, Bradley T. Hyman, Clifford R. Jack Jr., Claudia H. Kawas, William E. Klunk, Walter J. Koroshetz, Jennifer J. Manly, Richard Mayeux, et al. 2011. The diagnosis of dementia due to alzheimer’s disease: Recommendations from the national institute on aging-alzheimer’s association workgroups on diagnostic guidelines for alzheimer’s disease. *Alzheimer’s & Dementia*, 7(3):263–269.
- John C. Morris, Sandra Weintraub, Helena C. Chui, Jeffrey Cummings, Charles DeCarli, Steven Ferris, Norman L. Foster, Douglas Galasko, Neill Graff-Radford, Elaine R. Peskind, et al. 2006. The uniform data set (uds): clinical and cognitive variables and descriptive data from alzheimer disease centers. *Alzheimer Disease & Associated Disorders*, 20(4):210–216.
- Weerasak Muangpaisan, Chonachan Petcharat, and Varalak Srinonprasert. 2012. Prevalence of potentially reversible conditions in dementia and mild cognitive impairment in a geriatric clinic. *Geriatrics & Gerontology International*, 12(1):59–64.
- Kevin P. Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.
- Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffrey Kaye. 2011. Spoken language derived measures for detecting mild cognitive impairment. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(7):2081–2090.
- Vanessa Marquiefável Serrani. 2015. Ambiente web de suporte à transcrição fonética automática de lemas em verbetes de dicionários do português do brasil.
- Carlos N. Silla Jr. and Celso A.A. Kaestner. 2004. An analysis of sentence boundary detection systems for english and portuguese documents. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 135–141. Springer.
- Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.
- Camila Vieira Ligo Teixeira, Lilian Teresa Bucken Gobbi, Danilla Icassatti Corazza, Florindo Stella, José Luiz Riani Costa, and Sebastião Gobbi. 2012. Non-pharmacological interventions on cognitive functions in older people with mild cognitive impairment (mci). *Archives of gerontology and geriatrics*, 54(1):175–180.
- Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2).
- Ottokar Tilk and Tanel Alumäe. 2015. Lstm for punctuation restoration in speech transcripts. In *INTERSPEECH*.
- Xuancong Wang, Hwee Tou Ng, and Khe Chai Sim. 2012. Dynamic conditional random fields for joint sentence boundary and punctuation prediction. In *INTERSPEECH*.
- David Wechsler. 1997. *Wechsler memory scale (WMS-III)*. Psychological Corporation.
- Chenglin Xu, Lei Xie, Guangpu Huang, Xiong Xiao, Engsiong Chng, and Haizhou Li. 2014. A deep neural network approach for sentence boundary detection in broadcast news. In *INTERSPEECH*, pages 2887–2891.
- Maria Yancheva, Kathleen Fraser, and Frank Rudzicz. 2015. Using linguistic features longitudinally to predict clinical scores for alzheimer’s disease and related dementias. In *6th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, page 134.
- Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. 2002. The htk book. *Cambridge university engineering department*, 3:175.

Joint, Incremental Disfluency Detection and Utterance Segmentation from Speech

Julian Hough and David Schlangen

Dialogue Systems Group // CITEC // Faculty of Linguistics and Literature
Bielefeld University

firstname.lastname@uni-bielefeld.de

Abstract

We present the joint task of incremental disfluency detection and utterance segmentation and a simple deep learning system which performs it on transcripts and ASR results. We show how the constraints of the two tasks interact. Our joint-task system outperforms the equivalent individual task systems, provides competitive results and is suitable for future use in conversation agents in the psychiatric domain.

1 Introduction

Artificial conversational systems promise to be a valuable addition to the existing set of psychiatric health care delivery solutions. As artificial systems, they can ensure that interview protocols are followed, and, perhaps surprisingly, due to being “just a computer”, even seem to increase their interlocutors’ willingness to disclose (Lucas et al., 2014). Interactions with such conversational agents have been shown to contain interpretable markers of psychological distress, such as rate of filled pauses, speaking rate, and various temporal, utterance and turn-related interactional features (DeVault et al., 2013). Filled pauses and disfluencies in general have also been shown to predict outcomes to psychiatric treatment (Howes et al., 2012; McCabe et al., 2013).

Currently, these systems are only used to elicit material that is then analysed offline. For offline analysis of transcripts with gold standard utterance segmentation, much work exists on detecting disfluencies (Johnson and Charniak, 2004; Qian and Liu, 2013; Honnibal and Johnson, 2014). To enable more cost-effective analysis, however, and possibly even let the interaction script itself be dependent on an analysis hypothesis, it would be better to be able to work directly off the speech sig-

nal, and online (incrementally). This is what we explore in this paper, presenting and evaluating a model that works with online, incremental speech recognition output to detect disfluencies with various degrees of fine-grainedness.

As a second contribution, we combine incremental disfluency detection with another lower-level task that is important for responsive conversational systems, namely the detection of turn-taking opportunities through detection of utterance boundaries. (See for example (Schlangen and Skantze, 2011) for arguments for incremental processing and responsive turn-taking in conversational systems, and (Schlangen, 2006; Atterer et al., 2008; Raux, 2008; Manuvinakurike et al., 2016, *inter alia*) for examples of incremental utterance segmentation). Besides both being relevant for interactive health assessment systems, these tasks also have an immanent connection, as the approach typically used for turn-end detection is simply waiting for a silence of a certain duration, and hence is misled by intra-turn silent disfluencies. Similarly, without gold standard segmentation, disfluent restarts and repairs may be predicted at fluent utterance boundaries. We hence conjecture that the tasks can profitably be done jointly.

2 Related Work

As a separate task, there has been extensive work on utterance segmentation. Cuendet (2006) reports an NIST-SU utterance segmentation error rate result on the Switchboard corpus at 48.50, using a combination of lexical and acoustic features. Ang et al. (2005) report NIST-SU scores in the region of 34.35–45.92 on the ICSI Meeting Corpus. Martínez-Hinarejos et al. (2015) report state-of-the-art dialogue act segmentation results on Switchboard at 23.0 NIST-SU, however

this is not on the level of full dialogues, but on pre-segmented turn stretches. For the equivalent task of sentence boundary detection, Seeker et al. (2016) report an F-score of 0.7665 on Switchboard data, using a joint dependency parsing framework, and Xu et al. (2014) implement a deep learning architecture and report an 0.810 F-score and 35.9 NIST-SU error rate on broadcast news speech using prosodic and lexical features using a DNN for prosodic features, combined with a CRF classifier. However scaling this to spontaneous speech and the challenges of incrementality explained here, is yet to be tested.

Strongly incremental approaches to the task are rare, however (Atterer et al., 2008) achieve a word-by-word F-score of 0.511 on predicting whether the current word is the end of the utterance (dialogue act) on Switchboard, and using ground-truth syntactic information indicating sentence structure information achieve 0.559.

Disfluency detection on pre-segmented utterances in the Switchboard corpus has also had a lot of attention, and has also reached high performance (Johnson and Charniak, 2004; Georgila, 2009; Qian and Liu, 2013; Honnibal and Johnson, 2014). On detection on Switchboard transcripts, Honnibal and Johnson (2014) achieve 0.841 reparandum word accuracy using a joint dependency parsing approach, and Hough and Purver (2014) in a strongly incrementally operating system without look-ahead achieve 0.779, using a pipeline of classifiers and language model features. The potentially live approaches tend to use acoustic information (Moniz et al., 2015) and do not perform on a comparable level to their transcription-based task analogues, nor achieve the same fine-grained analysis of disfluency structure, which is often needed to identify the disfluency type and compute its meaning.

Live incremental approaches to both tasks have not been able to benefit from reliable ASR hypotheses arriving in a timely manner until recently. Now the arrival of improved performance, in terms of low Word Error Rate (WER) and better live performance properties is making this possible (Baumann et al., 2016). In this paper we define a joint task in a live setting. After defining the task we present a simple deep learning system which simultaneously detects disfluencies and predicts up-coming utterance boundaries from incremental word hypotheses and derived information.

3 The Tasks: Real-time disfluency prediction and utterance segmentation

3.1 Incremental disfluency detection

Disfluencies, in their fullest form as speech repairs, are typically assumed to have a tripartite *reparandum-interregnum-repair* structure (terms originally proposed by Shriberg (1994)), as exhibited by the following example.

$$\text{John } \underbrace{[\text{likes} + \{ \text{uh} \}]}_{\text{reparandum}} \underbrace{\text{loves}}_{\text{interregnum}} \text{] } \text{Mary} \quad (1)$$

If reparandum and repair are absent, the disfluency reduces to an isolated *edit term*. In the example given here, the interregnum is filled by a marked, lexicalised edit term, but more phrasal terms such as *I mean* and *you know* can also occur.

The task of disfluency detection then is to recognise these elements and their structure, and the task of *incremental* disfluency detection adds the challenge of doing this in real-time, from “left-to-right”. In that latter setting, detection runs into the same problem as a human processor of such an utterance: Only by the time the interregnum is encountered, or possibly even only when the repair is seen, does it become clear that earlier material now is to be considered as “to be repaired” (reparandum).¹ Hence, the task cannot be set up as a straightforward sequence labelling task where the tags “reparandum”, “interregnum” and “repair” are distributed left-to-right over words as indicated in the example above; in this example, it would unfairly require the prediction that “likes” is *going to be* repaired, at a point when no evidence is available for making it.

We follow Hough and Schlangen (2015) and use a tag set that encodes the reparandum start only at a time when it can be guessed, namely at the onset of the actual repair. This is illustrated in Figure 1 in the “disfluency (complex)” row. Here, the word at the repair onset, “to”, gets tagged as repair onset (*rpS*) and, at the same time, as repairing material beginning 5 tokens in the past (-5, yielding the complex label *rpS-5*). Additionally, we annotate all repair words (as *rpMid*, if the word is neither first nor last word of the repair, and together with the disfluency type, if it is the final word; here, the

¹Looking at it from a different perspective, this problem has been called the *continuation problem* by Levelt (1983): the repair material can only be integrated with the previous material, if it is identified as replacing the reparandum.

	A uh flight [to Boston + { uh I mean } to Denver] on Friday Thank you
Disfluency (simple)	<i>f e f f f e e e rpS f f f f f f</i>
Disfluency (complex)	<i>f e f f f e e e rpS-5 rpESub f f f f</i>
Utterance segmentation	<i>.w- -w- -w- -w- -w- -w- -w- -w- -w- -w- -w- .w- -w.</i>
Joint task (simple)	<i>.f- -e- -f- -f- -f- -e- -e- -e- -rpS- -f- -f- -f- .f- -f.</i>
Joint task (complex)	<i>.f- -e- -f- -f- -f- -e- -e- -e- -rpS-5- -rpESub- -f- -f- .f- -f.</i>

Figure 1: An utterance with the traditional repair disfluency and segmentation annotation in-line (Shriberg, 1994; Meteer et al., 1995) and our incrementally-oriented tag schemes

label is *rpESub* for substitution),² editing terms (*e*) and fluent material (*f*) as well. From the complex tag set, we can reconstruct the disfluency structure as in (1) in a strongly incremental fashion. We also define a reduced tag set (shown in Figure 1 as “disfluency (simple)”) that only tags fluent words, editing terms, and the repair onset.

3.2 Incremental utterance segmentation

We formulate incremental utterance segmentation as the judgement in real time as to when the current utterance is going to end, and so like (Schlangen, 2006; Atterer et al., 2008), we move from purely *reactive* approach, signalled by silence, to *prediction*. To allow prediction to be possible we use four tags for classifying stretches of acoustic data (which can be the time spans of forced aligned gold standard words, or the word hypotheses timings provided by an ASR), which are equivalent to a BIES (Beginning, Inside, End and Single) scheme for utterances— see Table 1.

The tag set allows evidence from the prior context of the word (the acoustic and linguistic information preceding the word) to be used to predict whether this word continues a current utterance (the *-* prefix) or starts anew (the *.* prefix), and also permits the online prediction of whether the next word (or segment) will continue the current utterance (the *-* suffix) or the current word ends the utterance (the *.* suffix). From these utterance boundary predictions can be derived when *-w.* or *.w.* is predicted (i.e. “will end utterance”). The tag set is summarized in Table 1 and an example is in Fig. 1, row “utterance segmentation”.

3.3 Defining the joint task

Studying the two phenomena in natural dialogue corpora, for example in terms of rich transcription mark-up in the SWBD annotation manual (Meteer et al., 1995), there are several constraints:

²The other repair type is delete *rpEDel*. Verbatim reparandum-repair repetitions are subsumed by *rpESub*.

	-w-	-w.	.w-	.w.
f	1	1	1	1
e	1	1	1	1
rpS	1	1	0	0

	-w-	-w.	.w-	.w.
f	1	1	1	1
e	1	1	1	1
rpS-[1-8]	1	0	0	0
rpMid	1	0	0	0
rpESub	1	1	0	0
rpEDel	1	1	0	0
rpS-[1-8]ESub	1	1	0	0
rpS-[1-8]EDel	1	1	0	0

Figure 2: The joint tag set for the task. 1= tag in set, simple (top) and complex (bottom).

- C1 Repair onsets cannot begin an utterance (by definition of first position repairs needing a preceding reparandum).
- C2 Repairs must be completed within the utterance in which they begin.
- C3 Utterances can be interrupted or abandoned, but these are different to within-dialogue-act repairs.

Given these constraints, we can generate a joint tag set as a subset of the cross product of both tag schemes. The utterance segmentation tags in Table 1 are combined with the simple strongly incremental disfluency tags described in §3.1. The joint set for both the simple and complex tasks is in Fig. 2, where 1 indicates the tag is in the set and 0 otherwise. In the simple task, there are 10 tags. The joint set for the full task including disfluency structure detection has 53 possible tags (rather than the full cross product, which would be 92). In reality, in the training corpus, only 43 of these possible combinations were found, so this constituted our tag set in practice. See Fig. 1 (bottom 2 rows) for example sequences.

3.4 Research questions

Given the formulation of the joint task, we would like to ask the following questions of scalable, automatic approaches to it:

-w-	a word which continues the current utterance and whose following word will continue it
-w.	a word which continues the current utterance and is the last word of it
.w-	a word which is the beginning of an utterance and whose following word will continue it
.w.	a word constituting an entire utterance

Table 1: The tag set for the continuity of each word within a dialogue act

- Q1 Given the interaction between the two tasks, can a system which performs both jointly help improve equivalent systems doing the individual tasks?
- Q2 Given the incremental availability of word timings from state-of-the-art ASR, to what extent can word timing data help performance of either task?
- Q3 To what extent is it possible to achieve a good online accuracy vs. final accuracy trade-off in a live, incremental, system?

To address these questions we use a combination of a deep learning architecture for sequence labelling and incremental decoding techniques which we will now explain.

4 LSTMs and Incremental Decoding for Live Prediction

Our systems consist of deep learning sequence models which consume incoming words and use word embeddings in addition to other features to predict disfluency and utterance segmentation labels for each word, in a strictly left-to-right, word-by-word fashion. We also use word timings as input to a separate classifier whose output is combined with that of the deep learning architecture in an incremental decoder. See Fig. 3 for the overall architecture. We describe the elements of the system below.

4.1 Input Features

In our systems we use the following input features:

- Words in a backwards window from the most recent word (transcribed or ASR)
- Durations of words in the current window (from transcription or ASR word timings)
- Part-Of-Speech (POS) tags for words in current window (either reference, or from an incremental CRF tagger)

For incremental ASR, we use the free trial version of IBM’s Watson Speech-To-Text service.³ The service provides good quality ASR on noisy

³<https://www.ibm.com/watson/developercloud/speech-to-text.html>

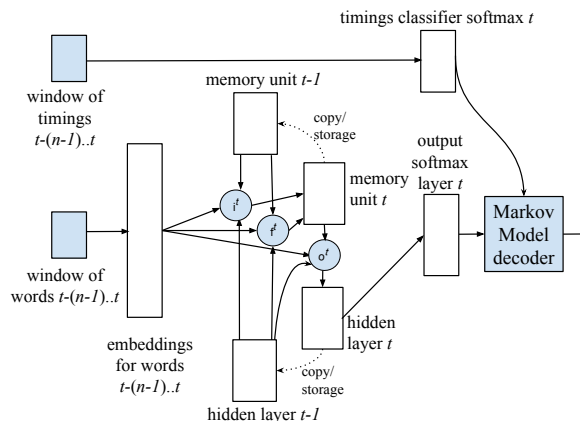


Figure 3: Schematic structure of the system.

data- on our selected heldout data on Switchboard, the average WER is 26.5%. The Watson service, crucially for our task, does not filter out hesitation markers or disfluencies, which is rare for current web-based services (Baumann et al., 2016). The service also outputs results incrementally, so silence-based end-pointing is not used. The service also returns word timings, which upon manual inspection were close enough to the reference timings to use as features in the live version of our system. In this paper, the durations are not features in the principal RNN but in an orthogonal logistic regression classifier- see §4.3.

For POS-tagging, we use the NLTK CRF tagger, which when trained on our training data and tested on our heldout data achieves 0.915 accuracy on all tags, which was sufficiently good for our purposes. Crucially, for the label UH , which is important evidence for an edit term, it achieves an F-score of 0.959.

4.2 Architectures

We use two well-studied deep learning architectures for our sequence labelling task- the Elman Recurrent Neural Network (RNN) and the Long Short-Term Memory (LSTM) RNN. Architecturally the RNNs here reproduce approximately the identical set-up as described in (Mesnil et al., 2013; Hough and Schlangen, 2015).

Input and word embeddings Following (Mes-

nil et al., 2013), we use 1-of-N, or ‘one-hot’, vectors as our raw input to the network, which provide unique indices to dense vectors in a word embedding matrix. The initial word embeddings were obtained from Switchboard data using the python implementation of `word2vec` in `gensim`,⁴ using a skip-gram context model. The training data for the initial embeddings was cleaned of disfluencies, effecting a ‘clean’ language model (Johnson and Charniak, 2004). These embeddings were then further updated as part of the objective function during the task-specific training itself. Instead of single word/POS inputs we use context windows which, like n-gram language models, are *backwards* from the current word. The internal representation of context windows of length n in the network is created through the ordered concatenation of the n corresponding word embedding vectors of size 50, resulting in an input to the network of dimension \mathbb{R}^{50n} . We use $n = 2$ in our experiments here.

RNN architecture and activation functions In addition to the embedding layer, we use a (recurrent) hidden layer of 50 nodes and an output layer the size of our training tag sets (43 nodes for the complex task and 10 nodes for the simple task). The standard Elman RNN dynamics in the recurrent hidden layer at time t is as in (3), where the hidden layer $h(t)$ is calculated as the Sigmoid function (2) of the addition of the weight matrix U' applied via dot product to the current input vector $x(t)$ and the weight matrix V' applied via dot product to the stored previous value of the hidden layer at time $t-1$, i.e. $h(t-1)$.

$$s(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

$$h(t) = s(U'x(t) + V'h(t-1)) \quad (3)$$

We use the standard softmax function for the node activation function of the output layer.

At decoding time, the compression of the context into the hidden layer allows us to save the current state of the decode live compactly from ASR results as they become available to the network. In order to integrate the new incoming words and POS tags with the history, it is only necessary to store the current hidden layer activation $h(t)$ (and the output softmax layer too, if that is being used by another process), and wait for new information to the input layer.

⁴<http://radimrehurek.com/gensim/>

LSTM unit In our LSTM, we include recurrent LSTM units that uses the input $x(t)$, the hidden state activation $h(t-1)$, and memory cell activation $c(t-1)$ to compute the hidden state activation $h(t)$ at time t . It uses a combination of a memory cell c and three types of gates: input gate i , forget gate f , and output gate o to decide if the input needs to be remembered (using the input gate), when the previous memory needs to be retained (forget gate), and when the memory content needs to be output (using the output gate). For each time step t the cell activations $c(t)$ and $h(t)$ are computed by the below steps, whereby the \odot is element-wise multiplication.

$$\begin{aligned} i(t) &= s(W'_i x(t) + U'_i h(t-1) + V'_i c(t-1)) \\ f(t) &= s(W'_f x(t) + U'_f h(t-1) + V'_f c(t-1)) \\ c(t) &= f(t) \odot c(t-1) + i(t) \odot \tanh(W'_c x(t) + U'_c h(t-1)) \\ o(t) &= s(W'_o x(t) + U'_o h(t-1) + V'_o c(t)) \\ h(t) &= o(t) \odot \tanh(c(t)) \end{aligned} \quad (4)$$

While many more weight matrices need to be learned (all the W' , U' and V' subscripted matrices), as with the standard RNN, at decoding time it is efficient to store the current decoding state in a compact way, as it is only necessary to save the activation of the memory cell $c(t)$ and the hidden layer $h(t)$ to save the current state of the network. See Fig. 3 for the schematic overall disfluency detection architecture for the LSTM.

Learning: error function and parameter update As is common for RNNs (De Mulder et al., 2015) we use negative log likelihood loss (NLL) as a cost function and use stochastic gradient descent over the parameters, including the embedding vectors, to minimize it. We use a batch size of 9 words, consistent with our repair tag scheme. Both networks use a learning rate of 0.005 and L2 regularisation on the parameters to be learned with a weight of 0.0001.

4.3 Incremental decoding and timing driven classifier

Markov model For decoding optimization we use Viterbi decoding on the sequence of softmax output distributions from the network in the spirit of (Guo et al., 2014). We use a Markov model which is hand-crafted to ensure legal tag sequences are outputted for the given tag set. In our joint task, this permits ‘late’ detection of an utterance boundary if the probability for a $-w.$ and following $.w-$ or $.w.$ tag on their own are not the arg max, but their combined probability permits the

best sequence. Similarly, in the complex task, repairs where evidence of a repair end tag is strong, but the repair onset tag was not the arg max can be detected at the repair end. From an incremental perspective, in Viterbi decoding there is the danger of output ‘jitter’. We investigate how different output representations have different effects on output prediction stability in our evaluation.

Timing driven classifier As an edition to the decoding step, we experimented with an independent timing driven classifier which consumes the durations of the last three words and outputs a probability that this is a fluent continuation or the beginning of a new utterance. We train a logistic regression classifier on our training data. Combining this two-class probability with the probability of the relevant utterance segmentation tags in decoding boosted performance considerably.

5 Evaluation Criteria

Accuracy On transcripts, we calculate repair onset detection accuracy F_{rPS} , where applicable reparandum word accuracy F_{rm} , and F1 accuracy for edit term words F_{e} , which includes interregna. For utterance segmentation we also use word-level F1 scores for utterance boundaries (end-of-utterance words) F_{uttSeg} . Carrying out the task live, on speech recognition hypotheses which very well may not be identical to the annotated gold-standard transcription, requires the use of time-based metrics of local accuracy in a time window (i.e. *within this time window, has a disfluency/utterance boundary been detected, even if not on the identical words?*)– we therefore calculate the F1 score over 10 second windows of each speaker’s channel. While this window-ing can give higher scores on certain phenomena, it tends to follow the word-level F-score so is a good time-based indicator of accuracy.

For utterance segmentation, for comparison to previous work we also use **NIST-SU** error rate (Ang et al., 2005). NIST-SU is the ratio of the number of incorrect utterance boundary hypotheses (missed boundaries and false positives) made by a system to the number of reference boundaries.

For a more coarse-grained metric which includes both tasks, which is useful in our target domain of interactions in a clinical context (Howes et al., 2014), we look at the **rpS : UttSeg ratio per speaker correlation** (Pearson’s R). This gives us the best approximation as to how good the system

is at estimating repair rate per utterance.

Timeliness and diachronic metrics Crucial for the live nature of the system, we measure latency (i.e. *how close to the actual time a disfluency or boundary event occurred has one been predicted?*) and also stability of output over time (i.e. *how much does the output change?*). For latency we use Zwarts et al. (2010)’s *time-to-detection* metric: the average distance (in numbers of words) consumed before first detection of gold standard repairs from the repair onset word, TD_{rPS} .⁵ We generalize this measure to the other tags of interest to give TD_{e} and $\text{TD}_{\text{uttSeg}}$ and also, particularly crucially for the ASR results, report the metrics in terms of time in seconds.⁶

For stability, incorporating insights from the evaluation of incremental processors by Baumann et al. (2011), we measure the **edit overhead (EO)** of the output labels– this is the percentage of unnecessary edits (insertions and deletions) required to get to the final labels outputted by the system.

6 Experimental Set-up

We experiment with the 2 joint output representations in Fig. 1 and implement an RNN and LSTM using Theano (Bergstra et al., 2010) as an extension to the code in Mesnil et al. (2013). We also run the 3 individual versions of the tasks with the tag sets shown in Fig. 1 for comparison. We also train a word timings driven classifier which adds information to the decoding step as explained above to try to answer Q2.⁷

Data We train on transcripts and test on both transcripts and ASR hypotheses. We use the standard Switchboard training data for disfluency detection (all conversation numbers beginning sw2*,sw3* in the Penn Treebank III release: 100k utterances, 650K words) and use the standard heldout data (PTB III files sw4[5-9]*: 6.4K utterances, 49K words) as our validation set. We test on the standard test data (PTB III files 4[0-1]*) with punctuation removed from all files.⁸ For

⁵Our measure is in fact one word earlier by default than Zwarts et al. (2010) as we take detection after the end of the repair onset word as the earliest possible detection point.

⁶These measures only apply to repairs and utterance boundaries detected correctly.

⁷All experiments are reproducible. The code can be downloaded at https://github.com/dsg-bielefeld/deep_disfluency

⁸We include partial words as these may in theory become available from the ASR in the live setting.

Eval. Method	System	F_{rm} (per word)	F_{rps} (per word)	F_{rps} (per 10s window)	F_e (per word)	F_e (per 10s window)	F_{uttSeg} (per word)	F_{uttSeg} (per 10s window)	NIST SU (word)	$rps / uttSeg$ / speaker correl.
Transcript	LSTM +timing	-	0.719	0.764	0.918	0.889	0.748	0.707	43.64	0.91
	LSTM	-	0.720	0.766	0.915	0.890	0.688	0.666	51.89	0.92
	LSTM(complex) +timing	0.601	0.693	0.730	0.91	0.888	0.707	0.685	50.07	0.82
	LSTM(complex) +timing	0.599	0.686	0.727	0.907	0.889	0.638	0.638	58.91	0.84
	RNN +timing	-	0.683	0.730	0.909	0.886	0.704	0.710	52.42	0.86
	RNN	-	0.685	0.728	0.908	0.884	0.647	0.635	57.75	0.87
	RNN(complex) +timing	0.572	0.663	0.715	0.908	0.882	0.699	0.669	50.89	0.83
	RNN(complex) +timing	0.568	0.659	0.713	0.905	0.882	0.621	0.613	60.74	0.81
ASR	LSTM +timing	-	-	0.551	-	0.727	-	0.685	-	0.72
	LSTM	-	-	0.548	-	0.726	-	0.630	-	0.79
	LSTM(complex) +timing	-	-	0.555	-	0.721	-	0.665	-	0.68
	LSTM(complex) +timing	-	-	0.557	-	0.721	-	0.601	-	0.67
	RNN +timing	-	-	0.542	-	0.718	-	0.681	-	0.69
	RNN	-	-	0.540	-	0.718	-	0.627	-	0.68
	RNN(complex) +timing	-	-	0.543	-	0.718	-	0.663	-	0.72
	RNN(complex) +timing	-	-	0.540	-	0.718	-	0.577	-	0.81

Table 2: Non-incremental (dialogue-final) results on transcripts and ASR results.

Eval. Method	System	F_{rps} (per word)	F_{rps} (per 10s window)	F_e (per word)	F_e (per 10s window)	F_{uttSeg} (per word)	F_{uttSeg} (per 10s window)	NIST SU (word)
Transcript	LSTM (uttSeg only)	-	-	-	-	0.727	0.679	46.17
	LSTM (disf only)	0.711	0.760	0.912	0.886	-	-	-
	LSTM (joint task)	0.719	0.764	0.918	0.889	0.748	0.707	43.64
ASR	LSTM (uttSeg only)	-	-	-	-	-	0.657	-
	LSTM (disf only)	-	0.531	-	0.721	-	-	-
	LSTM (joint task)	-	0.551	-	0.727	-	0.685	-

Table 3: Comparison of the joint vs. individual task performances

the ASR results evaluation, we only select a subset of the heldout and test data whereby both channels achieved below 40% WER to ensure good separation– this left us with 18 dialogues in the validation data and 17 dialogues for testing.

We train all RNNs for a maximum of 50 epochs else halt training if there is no improvement on the best F_{rm} score on the transcript validation set after 10 epochs.

7 Results and Discussion

Our dialogue-final accuracy results are in Table 2. On transcripts, our best per-word F_{rps} reaches 0.720 and best F_e reaches 0.918. For utterance segmentation, perword accuracy reaches 0.748 and the lowest NIST-SU error rate is 43.64. This is competitive with (Seeker et al., 2016)’s 0.767 F-score and out-performs (Cuendet, 2006) on the Switchboard data. The best $rps : uttSeg$ correlation per speaker reaches 0.92 ($p < 0.0001$).

In comparison to incremental approaches, we

outperform (Atterer et al., 2008)’s 0.511 accuracy on end-of-utterance. Their work allows no prediction lag in a strictly incremental setting, so is at a disadvantage, however our result of 0.748 on transcripts is reported alongside the average time to detection of 0.399 words, which suggests on average the uttSeg when predicted correctly, is done so with no latency.

With the exception of one metric, the LSTM outperforms the RNN on transcripts. The systems using the timing model in general outperform those with lexical information only on the utterance segmentation metrics, whilst not having an impact on disfluency detection.

According to the window-based accuracies, on ASR results there is significant degradation in accuracy for repair onsets (best F_{rps} =0.557) however utterance segmentation did not suffer the same loss, with the best system achieving 0.685 accuracy. The $rps : uttSeg$ Pearson’s R correlation per speaker reaches 0.81 ($p < 0.0001$) in a system with otherwise poor performance– the second

Eval. method	System	TTD_{rps} (word)	TTD_{rps} (time in s)	TTD_e (word)	TTD_e (time in s)	TTD_{uttSeg} (word)	TTD_{uttSeg} (time in s)	EO
Transcript	LSTM +timing	0.004	0.253	0.573	0.614	0.399	1.837	11.44
	LSTM	0.003	0.248	0.591	0.605	0.327	1.114	11.05
	LSTM(complex)	0.093	0.281	0.114	0.348	0.283	1.107	7.63
	LSTM(complex) +timing	0.090	0.293	0.135	0.483	0.369	1.960	8.51
ASR	LSTM +timing	-	0.202	-	0.734	-	3.247	20.71
	LSTM	-	0.199	-	0.649	-	1.645	20.44
	LSTM(complex)	-	0.236	-	0.341	-	2.303	20.70
	LSTM(complex) +timing	-	0.239	-	0.594	-	4.099	21.46

Table 4: Incremental results on transcripts and ASR results.

best achieved was 0.79 ($p < 0.0001$).

For disfluency detection, standard approaches use pre-segmented utterances to evaluate performance, so this result is difficult to compare. However in the simple task, the accuracy of 0.720 repair onset prediction is respectable (comparable to (Georgila, 2009)), and is useful enough to allow realistic relative repair rates, in line with our motivation. The complex tagging system performs poorly on repairs compared to the literature, however the lack of segmentation makes this a considerably harder task, in the same way as dialogue act tagging results are lower on unsegmented transcripts (Martínez-Hinarejos et al., 2015). Edit term detection performs very well at 0.918, approaching the state-of-the-art on Switchboard reported at 0.938 (Hough and Purver, 2014).

The utility of a joint task As can be seen in Table 3, the overall best performing systems on the individual tasks do not reach the results in any relevant metric of the best performing combined system. The disfluency-only systems were run ignoring all utterance boundary information, which puts this setting at a disadvantage to previous approaches, however it is clear that on unsegmented data our posing of the task jointly is useful.

Incrementality Incrementally the differences between the architectures was negligible— results for the LSTM are in Table 4. The latency for repair onset detection is very low, being detected as little as 0.196 seconds after the onset word is finished (or on transcripts largely directly after the word has been consumed as TTD_{rps} (word) = 0.003). Utterance boundaries were detected just over a second after the end of the last word of the previous utterance. However, the fact that TTD_{uttSeg} on the word level reaches 0.283 suggests the time-based average is being weighed down by occa-

sional long silences, which could be thresholded in future work. The EO measure of stability is severely affected by jittering ASR hypotheses, but given its worst result is 21.46% this is still a fairly stable incremental system.

Error Analysis To explore the errors being made by the systems, and how the RNN and LSTM may differ in ability, we performed an error analysis on the simple versions with the timing models— see Fig. 4. One can observe a boost in recall for various repair types in the LSTM, where it is performing better on repairs with longer reparanda. Characterizing repetitions as verbatim repeats, substitutions as the other repairs marked with a repair phase, and deletes as those without one, we see the LSTM outperforming the RNN on the rarer types. Whilst the problem is attenuated by the memory facility of the LSTM, our best system still suffers the vanishing gradient problem for predicting longer repairs with reparanda over 3 words long. Also we show in *uttSeg* detection all systems falter on long distance projections with coordinating conjunctions, which would potentially be dealt with more easily in a parsing framework, or a hierarchical deep learning framework.

We also investigated the *uttSeg* detection errors and see that the networks are generally not confusing disfluencies with boundaries. However, our best system incorrectly labelled 3.6% of the reference *uttSegs* as *rpS* (hence also affecting the precision of the *rpS* prediction)— upon inspection these were largely abandoned utterances, which according to the constraint C3 we posited above are not marked as disfluencies in the same way intra-utterance repairs are in the reference. Due to the original annotation instructions of (Meteer et al., 1995), these are segmented and not included in the traditional disfluency detection task. However,

		(a)	
Reparandum length	(support)	RNN recall %	LSTM recall %
1	(1487)	72.2	78.3
2	(477)	59.3	64.8
3	(155)	47.7	57.4
4	(73)	47.9	49.3
5	(31)	41.9	45.2
6	(15)	40.0	60.0
7	(9)	33.3	33.3
8	(4)	25.0	25.0

		(b)	
Repair Type	(support)	RNN recall %	LSTM recall %
repeat	(1043)	79.1	83.4
substitution	(1076)	59.2	66.4
delete	(132)	19.7	30.3

		(c)	
<i>uttSeg</i> Error Type		RNN % error	LSTM % error
FN, predicted <i>rpS</i>		2.9	3.6
FN, predicted <i>e</i>		4.9	4.4
FN, predicted CC		16.3	15.0
FN, predicted subj		7.0	6.5
FN, predicted proper		1.2	1.1
FN, predicted it		1.1	1.2
FN, predicted grounding		1.0	0.8
FN, predicted other		8.8	8.6
FN all		43.1	41.1
FP, predicted <i>uttSeg</i> for <i>rpS</i>		0.9	0.5
FP, predicted <i>uttSeg</i> for <i>e</i>		3.4	2.7
FP, predicted <i>uttSeg</i> for CC		5.1	3.6
FP, predicted <i>uttSeg</i> for subj		2.0	1.6
FP, predicted <i>uttSeg</i> for proper		0.9	0.6
FP, predicted <i>uttSeg</i> for it		0.5	0.4
FP, predicted <i>uttSeg</i> for grounding		0.7	0.4
FP, predicted <i>uttSeg</i> for other		6.9	2.7
FP all		20.4	12.5

Figure 4: Error analysis: (a) recall rates for *rpS* onsets of repairs with different reparandum lengths and (b) types, and (c) the source of errors in *uttSeg* detection.

intuitively these can be construed as a disfluency type, and in future we will treat them as a special type of *uttSeg*/disfluency hybrid.

As can be seen in Fig. 4 (c) other main sources of error are on coordinating conjunctions (CC) such as ‘and’ and ‘or’, nouns with nominative subject marking case like ‘I’ and ‘we’ (subj), other proper nouns, variants of ‘it’ and grounding utterances like ‘yeah’ and ‘okay’. *uttSeg* detection in both systems achieved high precision but relatively low recall.

8 Conclusion

We have presented the joint task of incremental utterance segmentation and disfluency detection and

show a simple deep learning system which performs it on transcripts and ASR results. As regards the research questions posed in §3.4, in answer to Q1, we showed that, all else being equal, a deep learning system can perform both tasks jointly improves over equivalent systems doing the individual tasks. In answer to Q2, we showed that word timing information, both from transcripts and ASR results, helps the utterance segmentation and the joint task across all settings whilst not aiding disfluency detection on its own, and in response to Q3, we achieve a good online accuracy vs. final accuracy trade-off in a live, incremental, system, however still experience some time delays for utterance segmentation in our most accurate system.

We conclude that our joint-task system for disfluency detection and utterance segmentation shows a new benchmark for the joint task on Switchboard data and due its incremental functioning on unsegmented data, including ASR result streams, it is suitable for live systems, such as conversation agents in the psychiatric domain. In future work we intend to optimize the inputs to our networks after this exploration, including using raw acoustic features, and combining the task with language modelling and dialogue act tagging.

Acknowledgments

We thank the EACL reviewers for their helpful comments. This work was supported by the Cluster of Excellence Cognitive Interaction Technology ‘CITEC’ (EXC 277) at Bielefeld University, funded by the German Research Foundation (DFG), and the DFG-funded DUEL project (grant SCHL 845/5-1).

References

- Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP (1)*, pages 1061–1064.
- Michaela Atterer, Timo Baumann, and David Schlangen. 2008. Towards incremental end-of-utterance detection in dialogue systems. In *COLING (Posters)*, pages 11–14.
- T. Baumann, O. Buß, and D. Schlangen. 2011. Evaluation and optimisation of incremental processors. *Dialogue & Discourse*, 2(1):113–141.
- Timo Baumann, Casey Kennington, Julian Hough, and David Schlangen. 2016. Recognising conversa-

- tional speech: What an incremental asr should do for a dialogue system and how to get there. In *International Workshop on Dialogue Systems Technology (IWSDS) 2016*. Universität Hamburg.
- James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.
- Sébastien Cuendet. 2006. Model adaptation for sentence unit segmentation from speech. Technical report, IDIAP.
- Wim De Mulder, Steven Bethard, and Marie-Francine Moens. 2015. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98.
- David DeVault, Kallirroi Georgila, and Ron Artstein. 2013. Verbal indicators of psychological distress in interactive dialogue with a virtual human. In *Proceedings of SigDial 2013*, pages 193–202.
- Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics.
- Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 554–559. IEEE.
- Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linguistics (TACL)*, 2:131–142.
- Julian Hough and Matthew Purver. 2014. Strongly incremental repair detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 78–89, Doha, Qatar, October. Association for Computational Linguistics.
- Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Proceedings of Interspeech 2015*, pages 849–853.
- Christine Howes, Matt Purver, Rose McCabe, Patrick GT Healey, and Mary Lavelle. 2012. Helping the medicine go down: Repair and adherence in patient-clinician dialogues. In *Proceedings of SemDial 2012 (SeineDial): The 16th Workshop on the Semantics and Pragmatics of Dialogue*, page 155.
- Christine Howes, Julian Hough, Matthew Purver, and Rose McCabe. 2014. Helping, i mean assessing psychiatric communication: An application of incremental self-repair detection. In *Proceedings of the 18th SemDial Workshop on the Semantics and Pragmatics of Dialogue (DialWatt)*, pages 80–89, Edinburgh, September.
- Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy-channel model of speech repairs. In *ACL*, pages 33–39.
- Willem J. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(4):41–104.
- Gale M. Lucas, Jonathan Gratch, Aisha King, and Louis Philippe Morency. 2014. It’s only a computer: Virtual humans increase willingness to disclose. *Computers in Human Behavior*, 37:94–100.
- Ramesh Manuvinakurike, Maike Paetzel, Cheng Qu, David Schlangen, and David DeVault. 2016. Toward Incremental Dialogue Act Segmentation in Fast-Paced Interactive Dialogue Systems. In *Proceedings of the 17th Annual SIGdial Meeting on Discourse and Dialogue*. Forthcoming.
- Carlos-D Martínez-Hinarejos, José-Miguel Benedí, and Vicent Tamarit. 2015. Unsegmented dialogue act annotation and decoding with n-gram transducers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):198–211.
- Rosemarie McCabe, Patrick GT Healey, Stefan Priebe, Mary Lavelle, David Dodwell, Richard Laugharne, Amelia Snell, and Stephen Bremner. 2013. Shared understanding in psychiatrist–patient communication: Association with treatment adherence in schizophrenia. *Patient education and counseling*, 93(1):73–79.
- Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775.
- M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer. 1995. Disfluency annotation stylebook for the switchboard corpus. ms. Technical report, Department of Computer and Information Science, University of Pennsylvania.
- Helena Moniz, Jaime Ferreira, Fernando Batista, and Isabel Trancoso. 2015. Disfluency detection across domains. In *The 6th Workshop on Disfluency in Spontaneous Speech (DiSS)*.
- Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of NAACL-HLT*, pages 820–825.
- Antoine Raux. 2008. *Flexible turn-taking for spoken dialog systems*. Ph.D. thesis, US National Science Foundation.

- David Schlangen and Gabriel Skantze. 2011. A General, Abstract Model of Incremental Dialogue Processing. *Dialogue & Discourse*, 2(1):83–111.
- David Schlangen. 2006. From reaction to prediction: Experiments with computational models of turn-taking. In *Proceedings of Interspeech 2006, Panel on Prosody of Dialogue Acts and Turn-Taking*.
- Anders Seeker, Agnieszka Björkelund, Wolfgang Falenska, and Jonas Kuhn. 2016. How to train dependency parsers with inexact search for joint sentence boundary detection and parsing of entire documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1923–1934, Berlin. ACL.
- Elizabeth Shriberg, Andreas Stolcke, Dilek Hakkani-Tür, and Gökhan Tür. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech communication*, 32(1):127–154.
- Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California, Berkeley.
- Chenglin Xu, Lei Xie, Guangpu Huang, Xiong Xiao, Engsiong Chng, and Haizhou Li. 2014. A deep neural network approach for sentence boundary detection in broadcast news. In *Proceedings of INTERSPEECH*, pages 2887–2891.
- Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1371–1378, Stroudsburg, PA, USA. Association for Computational Linguistics.

From Segmentation to Analyses: A Probabilistic Model for Unsupervised Morphology Induction

Toms Bergmanis

School of Informatics
University of Edinburgh
T.Bergmanis@sms.ed.ac.uk

Sharon Goldwater

School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

Abstract

A major motivation for unsupervised morphological analysis is to reduce the sparse data problem in under-resourced languages. Most previous work focuses on segmenting surface forms into their constituent morphs (e.g., *taking*: *tak* + *ing*), but surface form segmentation does not solve the sparse data problem as the analyses of *take* and *taking* are not connected to each other. We extend the MorphoChains system (Narasimhan et al., 2015) to provide morphological analyses that can abstract over spelling differences in functionally similar morphs. These analyses are not required to use all the orthographic material of a word (*stopping*: *stop* + *ing*), nor are they limited to only that material (*acidified*: *acid* + *ify* + *ed*). On average across six typologically varied languages our system has a similar or better F-score on EMMA (a measure of underlying morpheme accuracy) than three strong baselines; moreover, the total number of distinct morphemes identified by our system is on average 12.8% lower than for Morfessor (Virpioja et al., 2013), a state-of-the-art surface segmentation system.

1 Introduction

Most previous work on unsupervised morphological analysis has focused on the problem of **segmentation**: segmenting surface forms into their constituent morphs (Goldsmith, 2001; Creutz and Lagus, 2007; Poon et al., 2009; Lee et al., 2011; Virpioja et al., 2013; Sirts and Goldwater, 2013). However, the focus on surface segmentation is largely due to ease of model definition and implementation rather than linguistic correctness. Even in languages with primarily concatenative morphology, spelling (or phonological) changes often occur at

morpheme boundaries, so that a single morpheme may have multiple surface forms. For example, the past tense in English may surface as *-ed* (*walked*), *-d* (*baked*), *-ted* (*emitted*), *-ped* (*skipped*), etc.

A major motivation for unsupervised morphological analysis is to reduce the sparse data problem in under-resourced languages. While surface segmentation can help, the example above illustrates its limitations: for more effective parameter sharing, a system should recognize that *-ed*, *-d*, *-ted*, and *-ped* share the same linguistic function. The importance of identifying underlying morphemes rather than surface morphs is widely recognized, for example by the MorphoChallenge organizers, who in later years provided datasets and evaluation measures to encourage this deeper level of analysis (Kurimo et al., 2010). Nevertheless, only a few systems have attempted this task (Goldwater and Johnson, 2004; Naradowsky and Goldwater, 2009), and as far as we know, only one, the rule-based MORSEL (Lignos et al., 2009; Lignos, 2010), has come close to the level of performance achieved by segmentation systems such as Morfessor (Virpioja et al., 2013).

We present a system that adapts the unsupervised MorphoChains segmentation system (Narasimhan et al., 2015) to provide morphological analyses that aim to abstract over spelling differences in functionally similar morphemes. Like MorphoChains, our system uses an unsupervised log-linear model whose parameters are learned using contrastive estimation (Smith and Eisner, 2005). The original MorphoChains system learns to identify child-parent pairs of morphologically related words, where the child (e.g., *stopping*) is formed from the parent (*stop*) by adding an affix and possibly a spelling transformation (both represented as features in the model). However, these spelling transformations are never used to output underlying morphemes, instead the system just returns a segmentation by post-processing the inferred child-parent pairs.

We extend the MorphoChains system in sev-

eral ways: first, we use the spelling transformation features to output underlying morphemes for each word rather than a segmentation; second, we broaden the types of morphological changes that can be identified to include compounds; and third, we modify the set of features used in the log-linear model to improve the overall performance. We evaluate using EMMA (Spiegler and Monson, 2010), a measure that focuses on the identity rather than the spelling of morphemes. On average across six typologically varied languages (English, German, Turkish, Finnish, Estonian, Arabic), our system outperforms both the original MorphoChains system and the MORSEL system, and performs similarly to the surface segmentation system Morfessor. These results are (to our knowledge) the best to date from a system for identifying underlying morphemes; moreover, the total number of distinct morphemes identified by our system is on average 12.8% lower than for Morfessor, suggesting that it does a better job of abstracting over surface spellings and inducing a compact representation of the data.

2 Morphological Chains and Analyses

We base our work on the MorphoChains segmentation system (Narasimhan et al., 2015),¹ which defines a **morphological chain** as a sequence of **child-parent pairs**. Each pair consists of two morphologically related words where the child must be longer than the parent. To analyse a word we want to find the best parent for that word; we do so recursively until we conclude that the **stop condition** is met (i.e. a word doesn’t have a morphological parent). The word *standardizes*, for example, produces the following chain:

standardizes → *standardize* → *standard*

which consists of the child-parent pairs (*standardizes*, *standardize*), (*standardize*, *standard*) and (*standard*, NONE). Each child-parent pair is annotated with a **type** indicating the kind of **transformation** that relates the child-parent pair. The set of transformations defined by MorphoChains is: **suffixation** as in (*dogs*, *dog*), **prefixation** as in (*undone*, *done*), **deletion** as in (*baked*, *bake*)², **repetition** as in (*stopped*, *stop*), and **modification** as in

¹We modified the implementation available at <https://github.com/karthikncode/MorphoChain>.

²The system could in principle learn that *bake* is the parent of *baked* with type *suffix*, which would imply the analysis *bake* + *d*. However, we hope it learns instead the type *delete*, which implies the (correct) analysis *bake* + *ed*. Similar alternative analyses are possible for the other example types shown.

Word	MorphoChains	Our Model
stopping	stopp +ing	stop +ing
doubled	double +d	double +ed
acidified	acid +ifi +ed	acid +ify +ed

Table 1: Examples outputs of two models.

(*worried*, *worry*). We add a sixth type, **compound-ing** as in (*darkroom*, *room*). The delete, repeat, and modify types all assume the change occurs to the final character of the stem, while compounding can simply concatenate the two stems, or can introduce an extra character (as in *higher-rate* or German *schilddruese +n+ krebs* (‘thyroid cancer’).

Any word (*undone*) has many possible parents, some linguistically plausible (*undo*, *done*) and others not (*und*, *ndone*). Our system, like MorphoChains, learns a log-linear model to discriminate plausible from implausible parents amongst the complete **parent candidate set** for each word. The candidate set is generated by taking all possible splits of the word and applying all possible transformation types. For example, some parent candidates for the word *dogs* include (*dog*, suffix), (*do*, suffix), (*gs*, prefix), (*doga*, delete), (*dogb*, delete), (*doe*, delete), and (*doe*, modify). The last two imply analyses of *doe* + *gs* and *doe* + *s*, respectively.

The examples above indicate how analyses can be induced recursively by tracking the transformation type and orthographic change associated with each parent-child pair. However, the original MorphoChains algorithm did not do so, instead it only used the transformation types to predict morph boundaries. Table 1 contrasts the word segmentation into morphs produced by the original MorphoChains model and the morpheme analysis produced by our model. Table 2 provides additional examples of our recursive analysis process.

2.1 Model

We predict child-parent pairs using a log-linear model, following Narasimhan et al. (2015). The model consists of a set of features represented by a feature vector $\phi : \mathcal{W} \times \mathcal{Z} \rightarrow \mathbb{R}^d$, where \mathcal{W} is a set of words and \mathcal{Z} is the set of (*parent*, *type*) pairs for words in \mathcal{W} . The model defines the conditional probability of a particular (*parent*, *type*) pair $z \in \mathcal{Z}$ given word $w \in \mathcal{W}$ as:

$$P(z|w) = \frac{e^{\theta \cdot \phi(w,z)}}{\sum_{z' \in C(w)} e^{\theta \cdot \phi(w,z')}} , z \in C(w) \quad (1)$$

Step	Word	Child-Parent Pair	Type	Change	Analysis
1	acidified	(<i>acidified</i> , <i>acidify</i>)	modify	add <i>+ed</i>	<i>+ed</i>
2		(<i>acidify</i> , <i>acid</i>)	suffix	add <i>+ify</i>	<i>+ify +ed</i>
3		(<i>acid</i> , NONE)	stop	add <i>acid</i>	acid +ify +ed
1	doubled	(<i>doubled</i> , <i>double</i>)	delete	add <i>+ed</i>	<i>+ed</i>
2		(<i>double</i> , NONE)	stop	add <i>double</i>	double +ed
1	higher-rate	(<i>higher-rate</i> , <i>rate</i>)	compound	keep <i>higher</i>	
2		(<i>rate</i> , NONE)	stop	add <i>rate</i>	<i>rate</i>
3		(<i>higher</i> , <i>high</i>)	suffix	add <i>+er</i>	<i>+er rate</i>
4		(<i>high</i> , NONE)	stop	add <i>high</i>	high +er rate

Table 2: Examples of step-by-step derivations of morphological analysis of English words.

where $C(w) \subset \mathcal{Z}$ denotes the set of parent candidates for w and θ is a weight vector.

Our goal is to learn the feature weights in an unsupervised fashion. Following Narasimhan et al. (2015), we do so using Contrastive Estimation (CE) (Smith and Eisner, 2005). In CE every training example $w \in \mathcal{W}$ serves as both a positive example and a set of implied negative examples—strings that are similar to w but don’t occur in the corpus. The negative examples are the source of the probability mass allocated to the positive examples. The word w and its negative examples constitute the **neighbourhood** $\mathcal{N}(w)$ of w .

Given the list of words \mathcal{W} and their neighbourhoods, the CE likelihood is defined as:

$$L_{CE}(\theta, \mathcal{W}) = \prod_{w^* \in \mathcal{W}} \frac{\sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)}}{\sum_{w \in \mathcal{N}(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)}}. \quad (2)$$

We use the same neighbourhood functions as Narasimhan et al. (2015). Specifically, for each word w in the corpus \mathcal{W} , we create neighbours in two ways: by swapping two adjacent characters of w (*walking* \rightarrow *walkign*) and by swapping two pairs of adjacent characters, where one pair is at the beginning of the word, and the other at the end of the word (*walking* \rightarrow *awlkign*).

We use LBFGS-B (Zhu et al., 1997) to optimize the regularized log-likelihood of the model:

$$LL_{CE}(\theta, \mathcal{W}) = \sum_{w^* \in \mathcal{W}} \left[\log \sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)} - \log \sum_{w \in \mathcal{N}(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)} \right] - \lambda \|\theta\|^2 \quad (3)$$

2.2 Features

MorphoChains used a rich set of features from which we have kept some, discarded others and added new ones to improve overall performance. This section describes our set of features, with examples shown in Table 3.

Presence in Training Data We want features that signal which parents are valid words. Narasimhan et al. (2015) used each word’s log frequency. However the majority of words in the training data (word frequency lists) occur only once, which makes their frequency information unreliable.³ Instead, we use an out-of-vocabulary feature (OOV) for parents that don’t occur in the training data.

Semantic Similarity Morphologically related words exhibit semantic similarity among their word embeddings (Schone and Jurafsky, 2000; Baroni et al., 2002). Semantic similarity was an important feature in MorphoChains: Narasimhan et al. (2015) concluded that up to 25 percent of their model’s precision was due to the semantic similarity feature. We use the same feature here (COS). For a child-parent pair (w_A, w_B) with word embeddings v_{w_A} and v_{w_B} respectively we compute semantic similarity as:

$$\text{cos}(w_A, w_B) = \frac{v_{w_A} \cdot v_{w_B}}{\|v_{w_A}\| \|v_{w_B}\|} \quad (4)$$

Affixes Candidate pairs where the child contains a frequently occurring affix are more likely to be correct. To identify possible affixes to use as features, Narasimhan et al. (2015) counted the number of words that end (or start) with each substring

³The prevalence of singleton word types in the MorphoChallenge 2010 training data for English, German, Turkish ranges from 50.73 to 58.76 %.

No	Child	Parent Candid.	Active Features
1	<i>dog</i>	(<i>og</i> , prefix)	OOV, PREF= <i>d</i>
2	<i>decided</i>	(<i>decide</i> , delete)	DELETED= <i>e</i> , SUF= <i>ed</i> , SUFLIST
3	<i>decided</i>	(<i>decids</i> , delete)	OOV, DELETED= <i>s</i> , SUF= <i>ed</i> , SUFLIST
4	<i>stopped</i>	(<i>stop</i> , repeat)	REPEATED= <i>p</i> , RENV2= <i>op</i> , RENV1= <i>o</i> , SUF= <i>ed</i> , SUFLIST
5	<i>worried</i>	(<i>worry</i> , suffix)	MODIFIED= <i>y-i</i> , SUF= <i>ed</i> , SUFLIST
6	<i>higher-rate</i>	(<i>rate</i> , compound)	HEAD= <i>rate</i> , MODIFIER= <i>higher</i> , CONNECTOR= <i>-</i> , COMPOUND
7	<i>ratepayer</i>	(<i>payer</i> , compound)	HEAD= <i>payer</i> , MODIFIER= <i>rate</i> , COMPOUND
8	<i>decided</i>	(<i>deci</i> , compound)	OOV, HEAD= <i>deci</i> , MODIFIER= <i>ded</i>
9	<i>high</i>	(-, stop)	STOPLEN=4, STOPCOS=0.2
10	<i>decided</i>	(-, stop)	STOPLEN=7, STOPCOS=0.5
11	<i>unstable</i>	(<i>able</i> , prefix)	PREF= <i>unst</i>
12	<i>unable</i>	(<i>able</i> , prefix)	PREF= <i>un</i> , PREFLIST

Table 3: Examples illustrating which of the binary features in the model are active for various potential child-parent pairs. Not shown here is the real-valued semantic similarity feature COS, used in all examples except 9 and 10, where it is replaced by the binary feature STOPCOS= y , for y in increments of 0.1.

Prefixes
al, ar, ba, be, bo, ca, car, co, de, dis, en, ha, ho, in, inter, la, le, li, lo, ma, mar, mc, mi, mis, mo, out, over, pa, po, pre, pro, ra, re, ro, se, ta, to, un, under, up
Suffixes
a, age, al, an, ar, ary, as, ation, b, ble, ch, e, ed, el, en, er, ers, es, est, et, ful, i, ia, ic, ie, ies, in, ing, ings, is, ism, ist, ists, land, le, led, les, less, ley, ling, ly, m, man, ment, ments, ner, ness, o, or, p, s, se, son, t, ted, ter, ters, th, ting, ton, ts, y

Table 4: The likely English affixes found by using Letter Successor Entropy.

and selected the most frequent ones. However, all words that end with *ing* also end with *ng* and *g*, which means that they also become affix candidates. Furthermore, there are more words that end with *ng* or *g* than with *ing*, therefore valid affixes might be excluded from the list because of their more frequent substrings.

We therefore modify the affix features in two ways. First, we identify a more precise set of likely affixes using Letter Successor Entropy (LSE) values (Hafer and Weiss, 1974), which are typically high at morph boundaries. LSE is computed at each point in the word as the entropy of the distribution over the next character given the word prefix so far. When selecting likely affixes, we use an LSE threshold value of 3.0 as suggested by Hafer and Weiss (1974), and we require that the affix has appeared in at least 50 types with a cor-

pus frequency of at least 100. We then define two features (PREFLIST, SUFLIST), which are active if the proposed prefix or suffix for a parent-child pair is in the set of likely prefixes or suffixes. Table 4 shows the list of likely English affixes found by using LSE (62 suffixes and 42 prefixes). For German and Turkish, our other two development languages (see §3), the lists contain 498 suffixes/183 prefixes and 181 suffixes/35 prefixes, respectively.⁴

In addition, we use a much larger set of affix features, PREF= x and SUF= x , where x is instantiated with all possible word prefixes (suffixes) for which both w and xw (wx) are words in the training data.

Transformations To help distinguish between probable and improbable transformations, we introduce transformation-specific features. For deletion we use the deleted letter (DELETED). For repetition we use the repeated letter and its preceding 2- and 1-character contexts (REPEATED, RENV2 RENV1). For modification we use the combination of the involved letters (MODIFIED). Finally, for compounding we use the headword (i.e. the parent of the compound), the modifier and the connector, if such exists (HEAD, MODIFIER, CONNECTOR). Since these compound features can be very sparse, we also add a single COMPOUND feature, which is active when both parts of the compound are present in the training data.

Stop Condition To identify words with no parents we use two types of binary features suggested

⁴In MorphoChains, the number of affixes was set manually for each language tested: 300 for English, 500 for Turkish, and 100 for Arabic.

by Narasimhan et al. (2015). $\text{STOPCOS}=y$ is the maximum cosine similarity between the word and any of its parent candidates (using bins of size 0.1), and $\text{STOPLEN}=x$ is instantiated for all possible word lengths x in the training data. For illustration, if we are considering whether *decided* is a word with no parents (Table 3 Example 10), the binary features $\text{STOPLEN}=7$ and $\text{STOPCOS}=0.5$ become active.

We discard the starting and ending character unigram and bigram features used by MorphoChains, because of the large number⁵ and the sparsity of these features.

2.3 Data Selection

Most unsupervised morphology learners are sensitive to the coverage and the quality of training data. In a large corpus, however, many word types occur only once because of the Zipfian distribution of word types. Low-frequency types can be either rare but valid words or they can be foreign words, typos, non-words, etc. This makes learning from low-frequency words unreliable, but discarding them dramatically reduces the size of the training data (including many valid words).

To seek balance between the quality and the coverage of the training data we try to identify which low-frequency words are likely to provide useful statistical support for our model, so we can include those in the training data and discard the other low-frequency words. First, we set a frequency-based **pruning threshold** (PT) at the frequency for which at least 50% of the words above this frequency have a word embedding (see §3); next we set a **learning threshold** (LT) at the median frequency of the words with frequencies above PT; finally we adopt the algorithm by Neuvel and Fulop (2002) to decide which words with frequencies below PT can be useful to analyse the words with frequencies above LT. We filter out any remaining words with frequencies below PT.

The outline of the adapted version of the algorithm by Neuvel and Fulop (2002) is:

1) For every word pair in the top 20k most frequent words in training data:

1.1) We find the pair’s **orthographic similarities** as the longest common subsequence: *receive* ⇔ *reception*.

1.2) We find the pair’s **orthographic differ-**

ences with respect to their orthographic similarities: *receive* ⇔ *reception*.

2) For all word-pairs with the same orthographic differences we merge their similarities and differences into **Word Formation Strategies** (WFS): so *receive* ⇔ *reception*, *conceive* ⇔ *conception*, *deceive* ⇔ *deception* give *###ceive* ⇔ *###ception*, where * and # stand for the optional and mandatory character wild cards respectively.

3) We discard those WFS that that are suggested by less than 10 word pairs;

4) For each WFS and for each word with a frequency below PT:

4.1) if a word w matches either of the sides of a WFS and the other side of a WFS predicts a word w' with a frequency above the LT, we keep w in the training data, otherwise we discard it.

For more detailed description of the algorithm see Neuvel and Fulop (2002).

3 Experiments

Data We conduct experiments on six languages: Arabic, English, Estonian, Finnish, German and Turkish. For the word embeddings required by our system and the MorphoChains baseline, we used *word2vec* (Mikolov et al., 2013) to train a Continuous Bag of Words model on a sub-sample of the Common Crawl (CC) corpus⁶ for each language (Table 5 lists corpus sizes). We trained 100-dimensional embeddings for all words occurring at least 25 times, using 20 iterations and default parameters otherwise.

For all languages except Estonian, we train and evaluate all systems on the data from the Morpho Challenge 2010 competition.⁷ The training data consists of word lists with word frequencies. The official test sets are not public, but a small labelled training and development set is provided for each language in addition to the large unannotated word list, since the challenge included semi-supervised systems. Thus, for experiments on Arabic, English, Finnish, German and Turkish we evaluated on the annotated training and development gold standard analyses from the Morpho Challenge 2009/2010 competition data sets. The gold standard labels include part of speech tags and functional labels for inflectional morphemes, with multiple analyses given for words with part of speech ambiguity or

⁵For English there can be 676 different letter bigrams of which 99% occur at least once at the beginning of some word in the word frequency list.

⁶Common Crawl <http://commoncrawl.org>

⁷Morpho Challenge 2010: <http://research.ics.aalto.fi/events/morphochallenge2010>

Lang	Train (#types)	Test (#cases)	Embeddings (#tokens)
ARA	MC-10	MC-09	CC
	(19K)	(690)	(461M)
ENG	MC-10	MC-10	CC
	(878K)	(1569)	(1.78B)
EST	CC	S&G2013	CC
	(470K)	(1500)	(329M)
FIN	MC-10	MC-10	CC
	(2.9M)	(1835)	(1.18B)
GER	MC-10	MC-10	CC
	(2.3M)	(1779)	(856M)
TUR	MC-10	MC-10	CC
	(617K)	(1760)	(1.21B)

Table 5: Data statistics. MC-09/10: Morpho Challenge 2009/2010. CC: A sub-sample of Common Crawl. S&G2013: Sirts and Goldwater (2013)

functionally different but orthographically equivalent inflectional morphemes. For example, *rockiness* is analysed as *rock_N y_s ness_s*, while *rocks* has two analyses: *rock_N +PL* and *rock_V +3SG*.

For Estonian we train on word lists extracted from Common Crawl and test on data prepared by Sirts and Goldwater (2013). The Estonian test set contains only surface segmentation into morphs (e.g. *kolmandal* is analysed *kolmanda l*). Table 5 provides information about each dataset.

Since we are developing an unsupervised system, we want to make sure that it generalizes to new languages. We therefore divide the languages into three **development languages** (English, German, Turkish) and three **test languages** (Finnish, Estonian, Arabic). We used the development languages to choose features, design the data selection procedure and select best values for hyperparameters. The system that performed best on those languages was then used unmodified on the test languages.

Hyperparameters In addition to threshold values described above, we use the same $\lambda = 1$ (Equation 3) as Narasimhan et al. (2015). To control for under segmentation we downscale weights of the stop features by a factor of 0.8. We set the maximum affix length to 8 characters and the minimum word length to 1 character.

Evaluation Metric We test our model on the task of unsupervised morpheme analysis induction. We follow the format of Morpho Challenge 2010 and use Evaluation Metric for Morphological Analysis

(EMMA) (Spiegler and Monson, 2010) to evaluate predicted outputs. EMMA works by finding the optimal one-to-one mapping between the model’s output and the reference analysis (i.e., the spelling of the morphemes in the analysis doesn’t matter). These are used to compute precision, recall, and F-score against the reference morphemes.

Baselines We compare our model to three other systems: Morfessor 2.0 (Virpioja et al., 2013), MORSEL (Lignos et al., 2009; Lignos, 2010) and MorphoChains (Narasimhan et al., 2015). We also use a trivial baseline *Words* which outputs the input word.

Morfessor (*Morf.2.0*) is a family of probabilistic algorithms that perform unsupervised word segmentation into morphs. Since the release of the initial version of Morfessor, it has become popular as an automatic tool for processing morphologically complex languages.

MORSEL is a rule-based unsupervised morphology learner designed for affixal morphology. Like our own system, it outputs morphological analyses of words rather than segmentations. MORSEL achieved excellent performance on the Morpho Challenge 2010 data sets.

MorphoChains (*MC*) is the model upon which our own system is based, but as noted above it performs segmentation rather than analysis. In contrast to Morfessor and MORSEL, which analyse words based only on orthographic patterns, MorphoChains (like our extension) uses both orthographic and the semantic information.

All three baselines have multiple hyperparameters. Since performance tends to be most sensitive to the treatment of word frequency (including possibly discarding low-frequency words), for each system we tuned the hyperparameters related to word frequency to optimize average performance on the development languages, and kept these hyperparameters fixed for the test languages.

4 Results and Discussion

Table 6 gives the performance of all models on the three development languages. Our model outperforms all baselines on every language, and is a clear improvement over the original MorphoChains.⁸

⁸In the results reported by Narasimhan et al. (2015), MorphoChains appeared to outperform Morfessor, whereas we find the opposite. There are several possible reasons for the discrepancy. First, Narasimhan et al. (2015) used a segmentation-based metric rather than EMMA, so the scores are not comparable. Second, Narasimhan et al. (2015) appear to have tuned

Lang	Method	Prec	Recall	F-1
ENG	<i>Words</i>	0.750	0.362	0.489
	<i>Morf.2.0</i>	0.788	0.712	0.749
	<i>MORSEL</i>	0.784	0.725	0.752
	<i>MC</i>	0.685	0.729	0.706
	Our Model	0.787	0.741	0.763
GER	<i>Words</i>	0.776	0.258	0.387
	<i>Morf.2.0</i>	0.690	0.468	0.558
	<i>MORSEL</i>	0.670	0.449	0.538
	<i>MC</i>	0.649	0.397	0.492
	Our Model	0.590	0.548	0.568
TUR	<i>Words</i>	0.702	0.201	0.313
	<i>Morf.2.0</i>	0.598	0.338	0.432
	<i>MORSEL</i>	0.626	0.324	0.427
	<i>MC</i>	0.577	0.330	0.420
	Our Model	0.596	0.351	0.442
AVG	<i>Words</i>	0.743	0.274	0.396
	<i>Morf.2.0</i>	0.692	0.506	0.580
	<i>MORSEL</i>	0.693	0.449	0.572
	<i>MC</i>	0.637	0.485	0.539
	Our Model	0.658	0.547	0.591

Table 6: Results on development languages. Scores calculated using EMMA. *Words*=Trivial baseline which outputs the input word.

To see where the benefit is coming from, we performed ablation tests (Table 7). Results show the importance of the LSE-based affix features (Model-A). Using these features gives gains of +1.0%, 3.8% and 0.6% F-1 absolute on English, German and Turkish respectively over using the raw affix occurrence frequencies as used by Narasimhan et al. (2015). We can see that our data selection scheme (Model-D) is important for English (+3.5%) and German (+1.1%). Although we expected that the data selection scheme would have the biggest impact on Turkish because of its small training data, it has very little effect on this language. As expected, the compounding transformation (Model-C) has a prominent impact on German (+2.7%) and a modest effect on English and Turkish. The three features PREFLIST, SUFLIST, COMPOUND (Model-B) have the least impact on the model’s performance (on average 0.5% F-1 absolute), however the effect is substantial considering that this gain is achieved

their hyperparameters separately for each language. Finally, it is not clear how they tuned the frequency-related hyperparameters for Morfessor. We found that Morfessor performed better than MorphoChains when either low frequency words are pruned from its input, or its log-frequency option is used rather than raw frequency.

Lang	Method	Prec	Recall	F-1
ENG	Model-D	0.710	0.746	0.728
	Model-C	0.811	0.705	0.754
	Model-B	0.775	0.738	0.756
	Model-A	0.755	0.751	0.753
	Full Model	0.787	0.741	0.763
GER	Model-D	0.633	0.497	0.557
	Model-C	0.666	0.456	0.541
	Model-B	0.608	0.530	0.566
	Model-A	0.636	0.455	0.530
	Full Model	0.590	0.548	0.568
TUR	Model-D	0.554	0.365	0.440
	Model-C	0.601	0.344	0.438
	Model-B	0.604	0.344	0.437
	Model-A	0.588	0.346	0.436
	Full Model	0.596	0.351	0.442

Table 7: Ablation analysis. -D=no data selection, -C=no compound transformations, -B=no PREFLIST, SUFLIST, COMPOUND features, -A=no other LSE-based affix features.

by merely 3 features as opposed to a new feature type with many instantiations.

Table 8 shows some example outputs for English, German and Turkish. These analyses include some correctly identified spelling changes (Example 1) compounds (Example 4), and purely concatenative morphology (Example 6). In Example 2, *+ble* is counted as incorrect because our model predicts *+ble* both for *deplorable* and *reproducible* while the reference analysis uses *able_s* and *ible_s*, respectively. Since EMMA uses one-to-one alignment, it deems one of the alignments wrong. The opposite problem occurs in Example 4: our model analyses *aus* in two ways, either as a prefix *aus+* or as a separate word *aus* (part of a compound), whereas the reference analysis always treats it as a separate word *aus*. Example 6 illustrates an over-segmentation error, caused by encountering two similar forms of the verb *giy*, *giymeyin* and *giymeyi*.

Performance of all models on the three test languages is shown in Table 9. On average, our model does better than MorphoChains and MORSEL, but slightly worse than Morfessor. However, this difference is mainly due to Morfessor’s very good performance on Estonian, which is the only test set using gold standard segmentations rather than analyses. All systems perform poorly on Arabic since they do not handle templatic morphology; nevertheless our model and Morfessor perform considerably

No	Lang.	Test Word	Reference Analysis	Our Model
1	ENG	acknowledging	ac_p knowledge_N +PCP1	ac+ knowledge +ing
2	ENG	reproducible	re_p produce_V ible_s	re+ produce +ble
3	GER	wohnstuben	wohn_V stube_N +PL	wohn stube +n
4	GER	ausdrueckliche	aus drueck_V lich +ADJ-e	a <u>us</u> + drueck +lich +e
5	TUR	budaklara	budak +PL +DAT	budak +lar +a
6	TUR	giymeyin	giy +NEG_ma +P2_PL	giy +me +yi +n

Table 8: Examples morpheme analyses produced by our model on the development languages. Reference analyses in **bold** correspond to the predicted analyses that are incorrect. See text for further explanation.

Lang	Method	Prec	Recall	F-1
ARA	<i>Words</i>	1.000	0.112	0.202
	<i>Morf.2.0</i>	0.719	0.224	0.342
	<i>MORSEL</i>	0.993	0.135	0.238
	<i>MC</i>	0.988	0.125	0.221
	Our Model	0.839	0.206	0.331
FIN	<i>Words</i>	0.902	0.282	0.430
	<i>Morf.2.0</i>	0.704	0.389	0.501
	<i>MORSEL</i>	0.698	0.504	0.585
	<i>MC</i>	0.557	0.483	0.518
	Our Model	0.588	0.514	0.549
EST*	<i>Words</i>	0.951	0.572	0.715
	<i>Morf.2.0</i>	0.858	0.785	0.820
	<i>MORSEL</i>	0.686	0.777	0.729
	<i>MC</i>	0.840	0.611	0.707
	Our Model	0.756	0.763	0.760
AVG	<i>Words</i>	0.951	0.322	0.449
	<i>Morf.2.0</i>	0.760	0.466	0.554
	<i>MORSEL</i>	0.792	0.472	0.517
	<i>MC</i>	0.785	0.413	0.492
	Our Model	0.728	0.494	0.547

Table 9: Results on test languages. Scores calculated using EMMA. *=reference analysis contains word segmentation.

better than the others. Overall, our model performs consistently near the top even if not the best for any of the three languages.

Lexical Inventory Size One of the motivations for unsupervised morphological analysis is to reduce data sparsity in downstream applications, which implies that for a given level of accuracy, systems that produce a more compact representation (i.e., a smaller morpheme inventory) should be preferred. To see how compactly each model represents the test set, we count the number of unique morphemes (or morphs, or labels) in the predicted output of each model and compare it with the number of labels in the reference analysis and the num-

Method	ENG	GER	TUR	FIN	EST
<i>Morf.2.0</i>	1439	2005	1873	2586	1620
<i>MC</i>	1442	2004	1912	2718	1635
<i>MORSEL</i>	1373	1865	1748	2177	1562
<i>Ours</i>	1336	1600	1725	2114	1616
<i>Ref.Anal</i>	1257	1520	1361	1966	1548
<i>Words</i>	1569	1779	1760	1835	1500

Table 10: The number of distinct morphemes identified by each model. The number of distinct labels used in the reference analysis and the number of words in unanalysed test sets are given for comparison.

ber of words in the test set. Table 10 summarizes this information⁹. For all languages except Estonian our model finds the most compact set of items. The number of distinct morphemes identified by our model is only about 5%, 4.5% and 8.0% larger than in the reference analysis for English, German and Finnish respectively. On average our model identified 12.8% and 14.8% fewer morphemes than Morfessor and MorphoChains respectively, while on average performing no worse or better than the two word segmentation systems. MORSEL produces the second most compact output with only a 3.2% larger set of distinct morphemes than our model, leaving the two word segmentation systems, Morfessor and MorphoChains, in the third and the fourth place respectively. These results suggest that systems that attempt to output morphological analysis succeed in reusing the same morphemes more frequently than the systems that perform surface segmentation.

5 Conclusion

We presented an unsupervised log-linear model that learns to identify morphologically related words

⁹Arabic is excluded because of the low overall performance of all models (maximum recall 22.4%).

and the affixes and spelling transformations that relate them. It uses these to induce morpheme-level analyses of each word and an overall compact representation of the corpus. In tests on six languages, our system’s EMMA scores are considerably better than its inspiration, the segmentation system MorphoChains, and it also outperformed the rule-based analysis system MORSEL. Our system achieved similar EMMA performance to Morfessor but with a more compact representation—the first probabilistic system we are aware of to do so well. In future work, we hope to investigate further improvements to the system and perform extrinsic evaluation on downstream tasks.

References

- Marco Baroni, Johannes Matiassek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57. Association for Computational Linguistics, July.
- Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions of Speech and Language Processing*, 4(1):1–34, February.
- John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, June.
- Sharon Goldwater and Mark Johnson. 2004. Priors in Bayesian learning of phonological rules. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology*, pages 35–42, Barcelona, Spain, July. Association for Computational Linguistics.
- Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.
- Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho Challenge 2005-2010: Evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden, July. Association for Computational Linguistics.
- Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–9, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Constantine Lignos, Erwin Chan, Mitchell P. Marcus, and Charles Yang. 2009. A rule-based unsupervised morphology learning framework. In *CLEF (Working Notes)*.
- Constantine Lignos. 2010. Learning from unseen data. In Mikko Kurimo, Sami Virpioja, and Ville T. Turunen, editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38, Helsinki, Finland, September 2–3. Aalto University School of Science and Technology.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the 2013 International Conference on Representation Learning*.
- Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1531–1536.
- Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.
- Sylvain Neuvel and Sean A. Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 31–40. Association for Computational Linguistics, July.
- Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado, June. Association for Computational Linguistics.
- Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, pages 67–72, Lisbon, Portugal. Association for Computational Linguistics.
- Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using Adaptor Grammars. *Transactions of the Association for Computational Linguistics*, 1(May):231–242.
- Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 354–362, Ann Arbor, Michigan, June. Association for Computational Linguistics.

- Sebastian Spiegler and Christian Monson. 2010. EMMA: A novel evaluation metric for morphological analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1029–1037, Beijing, China, August. Coling 2010 Organizing Committee.
- Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.
- Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

Creating POS Tagging and Dependency Parsing Experts via Topic Modeling

Atreyee Mukherjee
Indiana University
atremukh@indiana.edu

Sandra Kübler
Indiana University
skuebler@indiana.edu

Matthias Scheutz
Tufts University
matthias.scheutz@tufts.edu

Abstract

Part of speech (POS) taggers and dependency parsers tend to work well on homogeneous datasets but their performance suffers on datasets containing data from different genres. In our current work, we investigate how to create POS tagging and dependency parsing experts for heterogeneous data by employing topic modeling. We create topic models (using Latent Dirichlet Allocation) to determine genres from a heterogeneous dataset and then train an expert for each of the genres. Our results show that the topic modeling experts reach substantial improvements when compared to the general versions. For dependency parsing, the improvement reaches 2 percent points over the full training baseline when we use two topics.

1 Introduction

POS tagging and dependency parsing perform well when trained and tested on datasets that are predominantly in the same text domain. However, there is decrease in accuracy for heterogeneous datasets, i.e., for datasets that consist of a mixture of data from different domains. Our current work focuses on improving POS tagging and dependency parsing for such heterogeneous datasets from a variety of different genres by creating experts for automatically detected topics. In our case, the datasets consist of newspaper reports on the one hand and biomedical extracts on the other.

For determining the topic of a sentence, we use Latent Dirichlet Allocation (LDA), which finds the latent topic structure in a document. In our case, a document to be clustered consists of a single sentence. We then assign each sentence to

the most likely topic, for both training and test sentences. We consequently train an expert for each topic and then use this expert to POS tag and parse the test sentences belonging to this topic. We assume that the topics detected by the topic modeler do not only pertain to lexical differences, which can be beneficial for the POS tagger and the parser, but also to syntactic phenomena. Thus, one topic may focus on “incomplete” sentences, such as headlines in a newspaper.

Our work is related to domain adaptation since the aim is to improve (morpho-)syntactic analysis for different domains. However, our approach can be regarded as a more general approach to the problem of domains as it is based on automatically determining the genres present in the dataset. Thus, no manual work is involved.

Our results show small to considerable improvements over a competitive baseline of using the full training set. For POS tagging, there is an improvement of 0.3 percent points over the full training set. For dependency parsing, the gain is more pronounced: almost 2% over the full training set.

The remainder of the paper is structured as follows: Section 2 discusses our research questions and section 3 the related work in the area. Section 4 describes the setup for our experiments, and section 5 shows the experimental results. We draw our conclusions in Section 6.

2 Research Questions

Our aim is to create POS tagging and parsing experts for heterogeneous datasets, with sentences from different genres. For example, the dataset might be a mixture of newspaper articles, blogs, financial reports, research papers and even specialized texts such as biomedical research papers and law texts. We create experts such that each

expert would learn specific information about its own genre. We determine these experts by performing topic modeling on sentences and then train an expert on the sentences of the topic. We group sentences based on their most probable topic. To test our hypothesis that topic modeling can serve to group sentences into topics, we create a mixed dataset from the financial domain (using the Penn Treebank (Marcus et al., 1994)) and from the biomedical domain (using the GENIA Corpus (Tateisi and Tsujii, 2004)) such that the new hand-crafted corpus consists of sentences from both domains in equal measure. Consequently, there is a clear difference in the genres in our corpus, and we have gold standard topic information.

We perform topic modeling on training and test data simultaneously: We assign a test sentence to the topic with the highest probability. This means that we currently simplify the problem of assigning new sentences to topics. In the future, we plan to assign new sentences to topics based their similarity to sentences in the topics created during training, following the work by Plank and van Noord (2011).

Our current research focuses on answering the following questions for POS tagging and parsing tasks:

Question 1: Does Topic Modeling Detect Topics?

In this question, we investigate whether an unsupervised topic modeler can detect topics in a heterogeneous corpus. We use our artificially created heterogeneous corpus containing sentences from the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1994) and from the GENIA Corpus (Tateisi and Tsujii, 2004) and take their original corpus as the gold standard topic. We assume that a good split into the known topics, financial news and biomedical abstracts, will also improve POS tagging and parsing accuracy. If we assume two topics, we should be able to see a clear distinction between WSJ and GENIA sentences. I.e., for each topic, we should have a clear correspondence of its sentences to either WSJ or GENIA. We thus calculate the percentage of sentences in a given topic that belong to GENIA and expect that one topic should have a high percentage and the other one a low percentage. We also experiment with a larger number of topics, to see if we can profit from a finer grained topic defini-

tion. However, this advantage will be offset by a smaller training set since we split into more sets.

Question 2: Does POS Tagging Benefit from Using Topics?

In this question, we examine whether the performance of POS tagging improves if we create experts based on the topics detected by the topic modeler. Thus, we use the topics created for the previous sections and train a POS tagging expert on the training part of each topic. We then use the expert to tag the test sentences from this topic. In this setting, we can see if the experts can effectively handle the data sparseness caused by dividing the training set into multiple experts. We experiment with one setting in which we use topic modeling as hard clustering, i.e., we assign each sentence to the topic for which the topic modeler gave the highest probability. We also experiment with soft clustering, in which we add each sentence to all topics, weighted by its probability distribution.

Question 3: Does Dependency Parsing Benefit from the Topics?

Here, we investigate the effects of using topic modeling experts for dependency parsing. We first use gold POS tags in order to abstract away from POS tagging quality. In a second step, we investigate the interaction between POS tagging and parsing experts. I.e., we are interested in whether dependency parsing can profit from using the POS tags that were determined by the POS tagging experts. This allows us to determine whether integrating POS information given by the POS experts can improve dependency parsing or whether there is no interaction between the two levels.

Question 4: What do the Experts Learn?

In this question, we will analyze the results from question 3 in more detail to investigate how the topic modeling experts improve parsing results. We are interested in whether there are specific types of sentences or dependencies that are grouped by the topic models, so that the parsing experts focus on a specific subset of syntactic properties.

3 Related Work

To the best of our knowledge, there is little direct correlation between our work on POS tagging and

parsing experts to that of the previous work done in the area. However, our work is comparable to domain adaptation since we create experts to tag and parse heterogeneous datasets. The work in this area is largely driven by the unavailability of examples from the target domain. Our work focuses on creating experts using topic modeling which will be able to tag and parse target domain sentences belonging to a specific topic. Compared to POS tagging, there has been significant work on domain adaptation in dependency parsing.

Dredze et al. (2007) found that problems in domain adaptation are compounded by differences in the annotation schemes between the treebanks. Blitzer et al. (2006) experimented with structural correspondence learning (SCL), which focuses on finding frequently occurring pivot features that occur commonly across domains in the unlabeled data but equally characterize source and target domains. Similar to our work, Blitzer et al. used the WSJ as the source and MEDLINE abstracts as the target domain. They established that SCL reaches better results in both POS tagging and parsing than supervised and semi-supervised learning even when there is no training data available in the target domain.

For POS tagging, Clark et al. (2003) applied an agreement-based and a baseline co-training method by using a Markov model tagger and a maximum entropy tagger. In case of the baseline, all the sentences from one tagger are added to train the other whereas in the agreement-based method, both taggers have to reach to the same decision for a sentence to be added to the training. Kübler and Baucom (2011) used a similar concept but with three different taggers and showed that selecting sentences as well as sequences of words for which all taggers agree yield the highest gains. Sagae and Tsujii (2007) emulate a single iteration of co-training by using MaxEnt and SVM, selecting the sentences where both models agreed and adding these sentences to the training set. Their approach reached the highest results on the domain adaptation task of CoNLL 2007 (Nivre et al., 2007).

In the CoNLL 2007 shared task on domain adaptation for dependency parsing, Attardi et al. (2007) used a tree revision method that corrects the mistakes caused by the base parser for the target domain. Later, Kawahara and Uchimoto (2008) employed a single parser approach using a second order MST parser and combining labeled

data from the known domain with unlabeled data of the new domain by simple concatenation and judging the efficacy of the resulting most reliable parses. Finkel and Manning (2009) devised a new model for named entity recognition as well as dependency parsing by using hierarchical Bayesian prior. This is influenced by the notion that different domains may have different features specific to each domain. However, instead of applying a constant prior over all the parameters, a hierarchical Bayesian global is used. This enables sharing of information across domains but also allows to override this information if there is ample evidence.

McClosky et al. (2010) designed the problem as “multiple source parse adaptation”, in which a parser was trained on multiple domains and learned the statistics as well as domain differences which affects the parser accuracy. Their parser outperformed the state-of-the-art baselines. This approach is similar to our work as we create experts based on topics, and each expert learns the specifics of the particular topic with which it is associated.

The closest approach to ours is the one by Plank and van Noord (2011), who employ a similar idea of using topic modeling for creating parsing experts. However, their task is to determine in a domain adaption setting which sentences of an out-of-domain training set are the most similar to the test set. Thus, they create a specialized training set for every document they need to parse while we create more general experts. In the work by Plank and van Noord (2011), the topic distribution in a document is used as features for their similarity metrics.

4 Experimental Setup

4.1 Data Sets

For our experiments, we use the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1994) and the GENIA Corpus (Tateisi and Tsujii, 2004). Both corpora use the Penn Treebank POS tagset (Santorini, 1990) with minor differences: The tagset used in GENIA is based on the Penn Treebank tagset, but it uses the tags for proper names and symbols only in very restricted contexts.

For the WSJ corpus, we extract the POS annotation from the syntactically annotated corpus. The GENIA Corpus comprises biomedical abstracts

from Medline, and it is annotated on different linguistic levels, including POS tags, syntax, coreference, and events, among others. We use GENIA 1.0 trees (Ohta et al., 2002) created in the Penn Treebank format¹. Both treebanks were converted to dependencies using `pennconverter` (Johansson and Nugues, 2007).

For our experiments, we need a balanced data set, both for the training and the test set. Since GENIA is rather small and since there is no standard data split for GENIA, we decided to extract the last 850 sentences for the test set. The remaining 17 181 sentences are used for training. For WSJ, we chose the same number of sentences for both training and the test set, the training sentences are selected randomly from sections 02-21 and the test sentences from section 22.

4.2 Topic Modeling

Probabilistic topic modeling is a class of algorithms which detects the thematic structure in a large volume of documents. Topic modeling is unsupervised, i.e., it does not require annotated documents (Blei, 2012) but rather discovers similarity between documents. Latent Dirichlet Allocation (LDA) is one of the topic modeling algorithms. It is a generative probabilistic model that approximates the underlying hidden topical structure of a collection of texts based on the distribution of words in the documents (Blei et al., 2003).

We use the topic modeling toolkit MALLET (McCallum, 2002). The topic modeler in MALLET implements Latent Dirichlet Allocation (LDA), clustering documents into a predefined number of topics. As a result, it provides different types of information such as:

- Topic keys: The highest ranked words per topic with their probabilities;
- Document topics: The topic distribution for each document (i.e., the probability that a document belongs to a given topic); and
- Topic state, which correlates all words and topics.

For our experiments, we use sentences as documents. Based on the document topic information, we then group the sentences into topics. We collect all sentences from the training and test set, cluster them via the MALLET topic modeler, and

determine for which expert(s) the sentence is relevant. There are several ways of determining the best expert, see below. Then, we separate the sentences for each expert into training and test sentences, based on the previously determined data splits (see above).

We can determine experts based on hard or soft clustering decisions: For hard clustering, the sentences are assigned to hard topics, based on the topic that has the highest probability in that sentence. I.e., if for sentence s_x , MALLET lists the topic t_1 as the topic with the highest probability, then s_x is added to the data set of topic t_1 . In other words, the data set of topic t_1 consists of all sentences for which MALLET showed topic t_1 as the most likely topic. This means that the data set sizes vary between topics.

For soft clustering experiments, we utilize the entire topic distribution of a sentence by weighting sentences in the training data based on their topic distribution. We simulate weighting training sentences by adding multiple copies to the training files of the experts. Thus, for 2-topic experiments, a sentence with 80% probability for topic 1 will be included 8 times in the expert for topic 1 and 2 times in the expert for topic 2, rounding up small percentages so that every sentence will be added to every expert at least once. Thus, we use a more fine grained topic model while mitigating data sparseness, but we risk adding non-typical / irrelevant sentences to experts.

4.3 POS Tagging

For part of speech tagging, we use the TnT (Trigrams'n'Tags) tagger (Brants, 2000). TnT is based on a second order Markov Model and has an elaborate model for guessing the POS tags for unknown words. We use TnT mainly because of its speed and because it allows the manual inspection of the trained models (emission and transition frequencies).

4.4 Dependency Parsing

For the parsing experiments, we used the dependency parser of the MATE Tools², a Java implementation of a graph-based parser (Bohnet, 2010).

4.5 Evaluation

We used the script `tnt-diff` that is part of TnT to evaluate the POS tagging results and the

¹<http://nlp.stanford.edu/mcclosky/biomedical.html>

²code.google.com/p/mate-tools

T.	2 topics		10 topics	
	% in train	%in test	% in train	% in test
1	0.71	0.71	0.48	0.52
2	97.99	98.6	98.58	98.35
3			1.16	0.73
4			94.87	97.14
5			0.17	0
6			0.28	0.29
7			99.47	99.12
8			98.93	100
9			98.92	99.33
10			94.85	95.35

Table 1: Distribution of sentences from the WSJ+GENIA data set given 2 and 10 topics (showing the percentage of GENIA sentences per topic).

CoNLL shared task evaluation script³ for evaluating the parsing results.

4.6 Baselines

We use two baselines: For the first baseline, we take the complete training set when no topic modeling is performed. Note that this is a very competitive baseline since the topic modeling experts have access to considerably smaller amounts of training data. In order to avoid differences in accuracy resulting from different training set sizes, we create a second baseline by splitting the sentences randomly into the same number of groups as the number of topics, while maintaining the equal distribution of WSJ and GENIA sentences. I.e., we assume the same number of random “topics”, all of the same size. Thus, in the 2-topic setting with the genres, we create two separate training sets, each containing half of the WSJ training set and half of the GENIA one. In this setting, we test all experts on the whole test set and average over the results.

5 Experimental Results

5.1 Does Topic Modeling Detect Topics?

Here we investigate whether LDA can separate the sentences into meaningful topics. Table 1 shows the distribution of sentences in the training and test set into different topics when we assume 2 or 10 topics. These results indicate that the topic modeler separates topics very efficiently. For the 2-

³<http://ilk.uvt.nl/conll/software/eval.pl>

Setting	Accuracy	
	2 topics	10 topics
Full training set	96.69	
Random split	96.41	95.48
Topic model	96.95	96.38
Soft Clustering	96.8	96.88

Table 2: Results of the POS tagging experiments.

topic experiments, a clear split is evident as the majority of the GENIA sentences are clustered in topic 2; the misclassified sentences constitute less than 1%. For the 10-topic experiments, we notice that topics 2, 4, 7, 8, 9, and 10 contain mainly GENIA sentences while the remaining topics cover mainly WSJ sentences. In both settings, the error rate is between 0.2% and 5%, i.e., we obtain a distinct split between GENIA and WSJ, which should give us a good starting point for the following experiments.

5.2 POS Tagging Experts

In this section, we investigate whether the POS tagger can benefit from using topic modeling, i.e., whether POS tagging results can be improved by training experts for genres provided by topic modeling. We compare the topic modeling approach to our two baselines for the 2-topic and 10-topic setting. We also perform a soft clustering experiment, in which each sentence is added to every topic, weighted by its probability (see section 4.2).

The results in Table 2 show that if we assume a 2-topic setting, the experts perform better than both baselines, i.e., the model trained on the full training set and the model with randomly chosen “topics”. The 2-topic expert model reaches an accuracy of 96.95%, which is slightly higher than the full training set accuracy of 96.69%. We know that the 2-topic setting provides a clear separation between WSJ and GENIA (Table 1). Thus, this setting outperforms the full training set using a smaller amount of training data. There is also an increase of 0.54 percent points over the accuracy of the 2 random split setting.

For the 10-topic setting, the topic expert model outperforms the random split of the same size by 0.9 percent points, which is a higher difference than for the 2-topic setting. This shows that the finer grained splits model important information. However, the topic expert model does not reach the accuracy of the baseline using the full training

Setting	LAS		UAS	
	2 topics	10 topics	2 topics	10 topics
Full training set	88.67		91.71	
Random split	87.84	84.91	90.86	88.64
Topic model	90.51	88.38	92.14	90.3
Soft clustering	89.86	89.91	91.99	91.84

Table 3: Results of the dependency parsing experiments using gold POS tags.

set. This can be attributed to the reduced size of the training set for the experts.

Since training set size is a detrimental factor for the larger number of topics, we also conducted an experiment where we used soft clustering so that every sentence is represented in every topic, but to a different degree. The last row in table 2 reports the results of this experiment. We notice that the 2-topic experts cannot benefit from the soft clustering. Since the separation between WSJ and GENIA is very clearly defined for the 2-topic experiments, the advantage of having a larger training set is outweighed by too many irrelevant examples from the other topic. However, the 10-topic model profits from the soft clustering, which indicates that soft clustering can alleviate the data sparseness problem of the POS tagging experts for larger numbers of topics. A more detailed analysis of the POS tagging results (on a slightly different data split), see (Mukherjee et al., 2016). This work includes an experiment showing that the POS tagging experts also increase performance for the WSJ corpus only, showing that POS tagging experts also perform better on more homogeneous collections, i.e., they adjust to less obvious differences between sentences.

5.3 Dependency Parsing Experts

5.3.1 Using Gold POS Tags

We now look into the parsing experiments using gold standard POS tags. The choice of gold POS tags allows us to focus on the contribution of the topic modeling experts on parsing results.

The results of the experiments are shown in Table 3, for 2-topic and 10-topic settings and in comparison to the two baselines, for the hard and soft clustering experiments. The hard clustering results indicate that the 2-topic expert model reaches an improvement over the baseline using the full training set for both the labeled attachment score (LAS) and the unlabeled attachment score (UAS). We find an increase of around 2% over the baseline

for LAS, and an increase of 0.43% for UAS. However, for the 10-topic setting, both the LAS and the UAS are slightly lower than the baseline. For LAS, the difference is 0.29 percent points while for UAS, the difference is 1.41 percent points. This shows that the gain in LAS and UAS is offset by the reduced training set, parallel to the results for POS tagging. Both the 2-topic and the 10-topic experts outperform the random split baseline (which uses similar training set sizes), with a gain of more than 3 percent points.

The soft clustering results show the same trends as in the POS tagging experiments: For the 2-topic setting, soft clustering outperforms the full baseline by 1.19 percent points. But it does not exceed the hard clustering results. In the 10-topic setting, soft clustering outperforms the full baseline as well as the hard clustering setting. This is because sentences with a 50% probability of belonging to topic 1 and a 40% probability for topic 3 need to be considered to belong to both topics. This result also shows that this method effectively handles the training data sparsity in the 10-topic setting.

5.3.2 Using the POS Tagger

In section 5.3, we use the gold standard POS tags in the POS tags. In this section, we explore the results of using POS tags from the POS tagger TnT as the input for the parser. This gives rise to four major scenarios:

1. The full training set is used for POS tagging and for parsing (full baseline).
2. Random splits are used for parsing and POS tagging. I.e., the POS tagger and parser are trained on random splits (random baseline).
3. Topic models are used for training the parser, but TnT is trained on the whole training set.
4. Topic models are used for training the parser and the POS tagger.

Setting	LAS		UAS	
	2 topics	10 topics	2 topics	10 topics
1. Full set POS + full set parsing	86.70		90.26	
2. Random split POS + random split parsing	85.77	81.33	89.11	85.73
3. Full set POS + topic model parsing	88.30	86.13	90.43	88.47
4. Topic model POS + Topic model parsing	88.35	85.68	90.55	88.15

Table 4: Results of the dependency parsing experiments using TnT POS tags.

Sentence	Fulltext	2-topic
	LAS	LAS
Phyllis Kyle, Stephenson Newport News , Va .	0	25.00
But volume rose only to 162 million shares from 143 million Friday .	46.15	61.54
Fidelity , for example , prepared ads several months ago in case of a market plunge .	47.06	82.35
CALL IT un-advertising .	50.00	75.00
(See related story : ” And Bills to Make Wishes Come True ” – WSJ Oct. 17 , 1989 .	52.38	61.90

Table 5: Comparison of LAS for the sentences with the lowest LAS in the fulltext setting.

We use the random split case as the lower baseline for these experiments and the full training set as the more competitive baseline. Table 4 shows the results.

Table 4 shows that in the 2-topic setting, using topic modeling experts on the POS level as well as on the parsing level reaches the highest results with an improvement of around 2% in LAS in comparison to the full baseline parser, from 86.70% to 88.35%. The gain in UAS is considerably smaller: The topic modeling expert reaches 90.55% as opposed to 90.26% for the full baseline. In contrast, the topic modeling setting for the 10-topic setting outperforms the random baseline but does not reach the full baseline, thus mirroring the trends we have seen before.

When we compare the experiments where we use the full POS tagging baseline along with topic model parsing experts (row 3 in table 4) to the full topic model (row 4), we see that the latter model reaches only very minimal gains by using the topic modeling POS tagger when we use 2 topics, and we have a negative trend when we use 10 topics. I.e. the overall quality of the POS tagger is more important than its specialization. Thus, even if the topic model POS tagger outperforms its full baseline, the learned adaptations only have a minimal effect on parsing accuracy.

5.4 Analysis of Results

We now have a closer look at the results presented for the parsing experiments using gold POS tags in section 5.3.1. The results show that the 2-topic parsing experts outperform the general parser trained on the full training set by almost 2 percent points. We looked at the 5 sentences that had the lowest LAS when we used the general parser. These sentences are shown in table 5, along with their LAS for both settings. The table clearly shows that the topic expert parsers reach a much higher LAS across all these sentences, and the highest increase reaches 35 percent points. We also see that there are two headlines among these sentences. They are different in their syntactic patterns from other sentences and thus difficult to parse. For this reason, we decided to have a closer at all “incomplete” sentences, i.e., sentences that do not have verbs, as an approximation of headlines. We found that of the 1 310 sentences in the training set, 437 were grouped into topic 1, the other 873 sentences in topic 2. In the test set, we had 65 such sentences, 15 in topic 1 and 50 in topic 2. For the sentences in topic 1, we calculate an LAS of 76.54, for the ones in topic 2 an LAS of 89.91. These results show that the parser expert for topic 2 has adapted substantially better to the syntax of such untypical sentences than the parser expert for topic 1.

We also looked at the dependency labels that were mislabeled most often by the more general,

Gold Dep.	Pred. Dep.	Fulltext	Topic 1	Topic 2
ADV	NMOD	121	37	86
PMOD	NMOD	101	21	67
NMOD	ADV	100	34	57
AMOD	NMOD	91	26	83
CONJ	NMOD	86	13	56

Table 6: The 5 most frequent dependency label confusions of the full baseline parser.

full baseline parser. The 5 most frequent combinations are shown in table 6, with their frequencies in the test sentences of the two topics. These numbers show that the topic 1 expert is much better adapted to these confusion sets, resulting in lower error rates than the topic 2. This shows very dramatically that the two topics learn different patterns.

6 Conclusion and Future Work

In these experiments, we have shown that we can improve parsing results on heterogeneous domains by using unsupervised topic modeling to separate the data into different topics. We can then train POS tagging and parsing experts on the individual topics, which show an increased accuracy in comparison to their counterparts trained on the whole, heterogeneous training set. We can mitigate the data sparsity resulting from having to split the training set into different topics by assigning every sentence to every topic but weighting their importance to a topic by the probabilities of the topic modeler. We also showed that while the POS tagger and the dependency parser individually profit from the split into topic experts, the combination of topic expert POS tagger and parser does not improve over using a POS tagger trained on the whole data set. We will have to investigate the reasons for this behavior in future work.

We will also investigate methods of how to assign test sentences to topics without having to re-run the topic modeler on the whole data set.

References

- Giuseppe Attardi, Felice Dell’Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1112–1118.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.
- David M. Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.
- John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 120–128, Sydney, Australia.
- Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 89–97, Beijing, China.
- Thorsten Brants. 2000. TnT—a statistical part-of-speech tagger. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics and the 6th Conference on Applied Natural Language Processing (ANLP/NAACL)*, pages 224–231, Seattle, WA.
- Stephen Clark, James Curran, and Miles Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL)*, Edmonton, Canada.
- Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1051–1055, Prague, Czech Republic.
- Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 602–610.
- Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia.
- Daisuke Kawahara and Kiyotaka Uchimoto. 2008. Learning reliability of parses for domain adaptation of dependency parsing. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, Hyderabad, India.

- Sandra Kübler and Eric Baucom. 2011. Fast domain adaptation for part of speech tagging for dialogues. In *Proceedings of the International Conference on Recent Advances in NLP (RANLP)*, Hissar, Bulgaria.
- Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop, HLT 94*, pages 114–119, Plainsboro, NJ.
- Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36. Association for Computational Linguistics.
- Atreyee Mukherjee, Sandra Kübler, and Matthias Scheutz. 2016. POS tagging experts via topic modeling. In *Proceedings of the 13th International Conference on Natural Language Processing (ICON)*, Varanasi, India.
- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL 2007 Shared Task. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 915–932, Prague, Czech Republic.
- Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 82–86, San Francisco, CA.
- Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1566–1576, Portland, OR.
- Kenji Sagae and Jun’ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, volume 2007, pages 1044–1050, Prague, Czech Republic.
- Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project. Department of Computer and Information Science, University of Pennsylvania, 3rd Revision, 2nd Printing.
- Yuka Tateisi and Jun’ichi Tsujii. 2004. Part-of-speech annotation of biology research abstracts. In *Proceedings of 4th International Conference on Language Resource and Evaluation (LREC)*, Lisbon, Portugal.

Universal Dependencies and Morphology for Hungarian – and on the Price of Universality

Veronika Vincze^{1,2}, Katalin Ilona Simkó¹, Zsolt Szántó¹, Richárd Farkas¹

¹University of Szeged
Institute of Informatics

²MTA-SZTE Research Group on Artificial Intelligence

kata.simko@gmail.com

{vinczev, szantozs, rfarkas}@inf.u-szeged.hu

Abstract

In this paper, we present how the principles of universal dependencies and morphology have been adapted to Hungarian. We report the most challenging grammatical phenomena and our solutions to those. On the basis of the adapted guidelines, we have converted and manually corrected 1,800 sentences from the Szeged Treebank to universal dependency format. We also introduce experiments on this manually annotated corpus for evaluating automatic conversion and the added value of language-specific, i.e. non-universal, annotations. Our results reveal that converting to universal dependencies is not necessarily trivial, moreover, using language-specific morphological features may have an impact on overall performance.

1 Introduction

Morphological tagging and syntactic parsing are key components in most natural language processing (NLP) applications. Linguistic resources and parsers for morphological and syntactic analysis have been developed for several languages, see e.g. the shared tasks on morphologically rich languages (Seddah et al., 2013; Seddah et al., 2014). However, the comparison of results achieved for different languages is not straightforward as most languages and databases apply a unique tagset, moreover, they were annotated following different guidelines. In order to overcome these issues, the project Universal Dependencies and Morphology (UD) has recently been initiated within the NLP community (Nivre, 2015). The main goal of the UD project is to develop a “universal”, i.e. a language-independent morphological and syntactic representation which can contribute to the im-

plementation of multilingual morphological and syntactic parsers from a computational linguistic point of view. Furthermore, it can enhance studies on linguistic typology and contrastive linguistics.

From the viewpoint of syntactic parsing, the languages of the world are usually categorized according to their level of morphological richness (which is negatively correlated with configurationality). At one end, there is English, a strongly configurational language while there is Hungarian at the other end of the spectrum with rich morphology and free word order (Fraser et al., 2013). In this paper, we present how UD principles were adapted to Hungarian, with special emphasis on Hungarian-specific phenomena.

Hungarian is one of the prototypical morphologically rich languages thus our UD principles can provide important best practices for the universalization of other morphologically rich languages. The UD guidelines for Hungarian were motivated by both linguistic considerations and data-driven observations. We developed a converter from the existing Szeged Dependency Treebank (Vincze et al., 2010) to UD and manually corrected 1,800 sentences from the newspaper domain. The experiences gained during the converter development and during the manual correction could reinforce the linguistic guidelines. Moreover, the manually corrected gold standard corpus provides the opportunity for empirical evaluations like assessing the converter and comparing dependency parsers employing the original and the universal morphological representations. Thus, we evaluated the quality of the automatic conversion, which reveals that converting to universal dependencies is not necessarily trivial, at least for Hungarian. We also show that using different morphological tagsets may have an impact on overall parsing performance and utilizing language-specific, i.e. non-universal, information has a considerable

added value at both the morphological and syntactic layers.

The chief contributions of the paper are the introduction of

- the universal morphology and dependency principles for Hungarian, leading to insights for other morphologically rich languages,
- empirical experiments on the upper bound of the accuracy of automatic conversion and pre-parsing,
- comparative evaluations for assessing the added value of language-specific information at the morphological and syntactic layers along with the interaction of these two.

2 Related Work

Standardized tagsets for both morphological and syntactic annotations have been constantly developed in the international NLP community. For instance, the MSD morphological coding system was developed for a set of Eastern European languages (Erjavec, 2012), within the MULTTEXT-EAST project. Interset functions as an interlingua for several morphological coding systems, which can convert different tagsets to the same morphological representation (Zeman, 2008). There have also been some attempts to define a common set of parts-of-speech: Rambow et al. (2006) defined a multilingual tagset for part-of-speech (POS) tagging and parsing, while McDonald and Nivre (2007) identified eight POS tags based on data from the CoNLL-2007 Shared Task (Nivre et al., 2007). Petrov et al. (2012) offered a tagset of 12 POS tags and applied this tagset to 22 languages.

Now, Universal Dependencies (UD) is the latest standardized tagset that we are aware of. UD is an international project that aims at developing a unified annotation scheme for dependency syntax and morphology in a language-independent framework (Nivre, 2015). Hungarian was among the first 10 languages of the project, participating also in the first official release in January 2015. In the latest release (Version 1.3, May 2016), there are annotated datasets available for 40 languages, including English, German, French, Hungarian and Irish, among others¹. In these datasets, the very same tagsets are applied at the morphological and

syntactic levels and texts are annotated on the basis of the same linguistic principles, to the widest extent possible.

The UD tagset encodes morphological information in the form of POS tags and feature–value pairs. As for syntactic information, each word is assigned to its parent word in the dependency tree and the grammatical function of the specific word is encoded in dependency labels. Dependency labels, POS tags and features are universal (i.e. there is a fixed set of them without the possibility of introducing new members), but values and dependency labels can have language-specific additions if needed. Features are divided into the categories lexical features and inflectional features. Lexical features are features that are characteristics of the lemmas rather than the word forms, whereas inflectional features are those that are characteristics of the word forms. Both lexical and inflectional features can have layered features: some features are marked more than once on the same word, e.g. a Hungarian noun may denote its possessor’s number as well as its own number. In this case, the Number feature has an added layer, Num[psor].

Up to now, several papers have been published on the general principles behind UD (Nivre, 2015; Nivre et al., 2016) or on specific treebanks. For instance, there are UD treebanks available for agglutinative languages such as Finnish (Haverinen et al., 2014; Pyysalo et al., 2015), Estonian (Muischnek et al., 2016) and Japanese (Tanaka et al., 2016), for Slavic languages (Zeman, 2015) and spoken Slovenian (Dobrovoljc and Nivre, 2016) and for Nordic languages such as Norwegian (Øvrelid and Hohle, 2016), Danish (Johannsen et al., 2014) and Swedish (Nivre, 2014), together with several other languages (Persian (Seraji et al., 2016) and Basque (Aranzabe et al., 2014), just to name a few). Recently, a further extension on the UD relations has been proposed: enhanced English dependencies are described in Schuster and Manning (2016).

Our UD principles introduced in this paper follow the central UD guidelines (Nivre, 2015) and we did our best to align with the existing guidelines for other morphologically rich languages as well. On the other hand, there are several Hungarian-specific phenomena that required changes and extensions of the original UD principles.

The only available manually annotated tree-

¹<http://universaldependencies.org/>

bank for Hungarian is the Szeged Corpus (Csendes et al., 2004) and Szeged Dependency Treebank (Vincze et al., 2010). It contains approximately 82,000 sentences and 1.5 million tokens, all manually annotated for POS-tagging and constituency and dependency syntax. We developed an automatic tool that converts the morphological descriptions of the Szeged Corpus to universal morphology tags and the dependency trees of the Szeged Treebank to universal dependencies.

3 Universal Morphology for Hungarian

In this section, we present the morphological tagset applied to Hungarian.

When adapting the principles of Universal Morphology to Hungarian, we were able to automatically convert most of the morphological features used in the Szeged Treebank 2.5 (Vincze et al., 2014), which was based on MSD principles (Erjavec, 2012). However, we faced some problematic issues, which we will discuss in detail in this section. The details of universal morphological codeset of Hungarian are available on our website².

3.1 Possessive constructions

The possessor in Hungarian possessive constructions can have two different surface forms, without any difference in meaning: the possessor can be morphologically marked or not, just like the English constructions *the girl's doll* and *the doll of the girl*. Thus, both of the following possessive constructions are widely used:

- (1) a szomszéd kertje
the neighbor garden-3SGPOSS
the neighbor's garden
- (2) a szomszédnak a kertje
the neighbor-DAT the garden-3SGPOSS
the neighbor's garden

In Example 1, the possessor is not marked, i.e. it shares its form with the nominative form of the noun, however, in Example 2, the possessor is morphologically marked, sharing its form with the dative form of the noun. Nevertheless, the possessed is morphologically marked in both cases, which was a novelty in the UD project as the languages already included in the data do not mark

²<http://rgai.inf.u-szeged.hu/project/nlp/research/msdkr/univmorph.html>

the possessor on the possessed noun but use determiners for this purpose (cf. *my car* but *az autóm* (the car-1SGPOSS)). Moreover, the number of the possessed can be marked on the noun in elliptical constructions such as:

- (3) Láttam az
see-PAST-2SGPOSS-ACC the
autódat , de a
car-2SG-POSS , but the
szomszédét nem .
neighbor-POSSD.SG-ACC not
I could see your car but not that of the neighbor.
- (4) Láttam a
see-PAST-2SGPOSS-ACC the
gyerekeidet , de a
child-2SG-PL-POSS , but the
szomszédéit nem .
neighbor-POSSD.PL-ACC not
I could see your children but not those of the neighbor.

Hence, we had to introduce novel morphological features to mark the person and number features of the possessor on Hungarian nouns. Number denotes the number of the noun, Number[psor] and Person[psor] denote the number and person of the possessor, and Number[psed] denotes the number of the possessed. Below, there is a sample word annotated according to the Universal Morphology principles.

- (5) házaiménak
house-1SGPOSS-PL-POSSD.SG-DAT
to that of my houses
NOUN
Case=Dat|Number=Plur|Number[psed]=Sing
|Number[psor]=Sing|Person[psor]=1

3.2 Object-verb agreement

Another Hungarian-specific feature was the definiteness of the object. As a special type of agreement, the definiteness of their objects determines which paradigm of the verb is to be chosen. In other words, the form of the verb changes when the definiteness of the object also changes (Törkenczy, 2005). For instance, proper nouns and NPs with a definite article are typical examples of definite objects and trigger the objective form of the verb (see Example 6) while bare nouns and NPs with an indefinite article are indefinite objects

(see Example 7) and trigger the subjective form of the verb. Second person objects also trigger a special form of the verb as listed in Example 8:

- (6) Látom Pistit .
see-1SGOBJ Steve-ACC .
I can see Steve.
- (7) Látok egy gyereket az udvaron .
see-1SGSUBJ a kid-ACC the yard-SUP .
I can see a kid in the yard.
- (8) Látlak .
see-1SGOBJ2 .
I can see you.

In this way, the feature Definiteness needs to be applied to verbs in Hungarian, moreover, it has a language-specific feature due to the special form triggered by the second person objects. Thus, Definiteness has three possible values in Hungarian: Definite, Indefinite, 2.

3.3 Determiners and pronouns

Determiners, pronouns and ordinal numbers also constituted a peculiarity. According to Hungarian grammatical traditions, ordinal numbers have been treated as numerals but in the universal morphology, they have to be annotated as adjectives. Thus, their POS tags were automatically converted to adjectives.

Demonstrative pronouns were also treated differently in the original annotation used in the Szeged Treebank and in universal morphology. While demonstrative pronouns *ez* and *az* are tagged as pronouns independently of their positions, in universal morphology such words occurring before an article should be tagged as a determiner (see Example 9) but when they are used as an NP, they should be tagged as a pronoun (see Example 10).

- (9) Olvastam azt a könyvet .
read-PAST-1SGOBJ that-ACC the book-ACC .
I have read that book.
- (10) Olvastam azt .
read-PAST-1SGOBJ that-ACC .
I have read that.

These cases were also automatically converted, following the universal morphology guidelines.

3.4 Verbal prefixes

In our original treebank, verbal particles that were spelt as a separate token had their own part-of-speech, i.e. verbal particle. According to the UD description however, not all function words that are traditionally called particles automatically qualify for the PART tag. They may be adpositions or adverbs by origin, therefore should be tagged ADP or ADV, respectively. Thus, we manually compiled a list that contained the original part-of-speech of words that were tagged as verbal prefixes, for instance, *el* “away” was treated as an adverb and *agyon* brain-SUP as a noun – the latter is usually used in phrases like *agyonüt* “kill someone by hitting on his head”. Based on this list, we were able to automatically assign UD POS tags to verbal prefixes.

4 Universal Dependency in Hungarian

When adapting the universal dependency labels to Hungarian, we could find a one-to-one correspondence between the original labels of the Szeged Treebank and the UD labels only in most of the cases, and these labels could be automatically converted to the UD format, making use of the dependency and morphological annotations found in the original treebank. However, we encountered some problematic cases during conversion, which we will discuss below in detail. The details of universal dependency rules of Hungarian are available on our website³.

4.1 Non-overt copulas

Traditionally, it is the verb that functions as the head of the clause in dependency grammars but in certain languages, there are verbless clauses where the predicate consists of a single nominal element (typically a noun or an adjective) at the surface level. The dependency analysis of such sentences may be problematic due to the lack of an overt verb. Some studies such as Polguère and Mel’čuk (2009) argue for a zero copula in such cases, especially when the copula is empty only in certain slots of the verbal paradigm. For instance, in Hungarian, the copula has its zero form only in the present tense, indicative mood, third person forms as shown in Examples 11-14:

- (11) Present tense, indicative mood, Sg1:

³<http://rgai.inf.u-szeged.hu/dependency>

Én tanár vagyok .
I teacher be-1SG .

I am a teacher.

(12) Present tense, indicative mood, Sg3:

Ő tanár .
he teacher .

He is a teacher.

(13) Past tense, indicative mood, Sg3:

Ő tanár volt .
he teacher be-PAST-3SG.

He was a teacher.

(14) Present tense, imperative mood, Sg3:

Ő legyen tanár !
he be-IMP-3SG teacher !

He should be a teacher.

The original dependency analysis in the Szeged Treebank inserts a zero copula (VAN), i.e. a virtual node in the dependency tree, which functions as the head of the clause and the nominal predicate is attached to it. Figure 1 shows such an analysis of the sentence *E gondolat sem új* (this thought not new) “This thought is not novel at all”.

Beside the function head analysis (i.e. where function words, e.g. the copula is the head), there is another approach to dependencies, namely, the content head analysis, where the head is a content word instead of a function word. In the latter case, the main grammatical relations can be found among content words and all the other function words are attached to the main structure. UD applies the content head analysis, which means that in copular constructions, the nominal element is the head and the copula (if present) is attached to it with a `cop` relation. In a similar way, the head of adpositional constructions is the noun and the adposition is attached to it.

Sentences with nominal predicates were automatically converted from the original treebank into the UD format: Figure 2 shows the UD analysis of the sentence found in Figure 1. Likewise, postpositional constructions were converted: the noun was treated as the head and the postposition was attached to it with a `case` label.

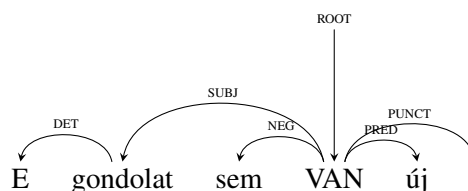


Figure 1: A function head analysis in the Szeged Dependency Treebank (*E gondolat sem VAN új* (this thought IS not new) “This thought is not novel at all”).

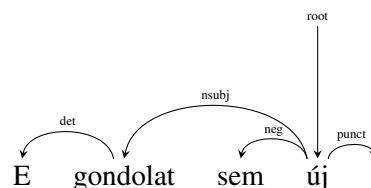


Figure 2: A content head analysis in the Hungarian UD treebank (*E gondolat sem új* (this thought not new) “This thought is not novel at all”).

4.2 Subordinate clauses

Subordinate clauses proved also to be a problematic issue as UD principles make a sharp distinction among several types of subordinate clauses – e.g. clausal subject, clausal object, adverbial clause – in contrast with the Szeged Dependency Treebank, which applies one single label for all types of subordinate clauses. Some types of subordinate clauses had a special label in the constituency version of the treebank hence their conversion was straightforward. In other cases, we could rely on manually constructed conversion rules but the resulting trees had to be corrected manually.

4.3 Multiword named entities

The UD treatment of multiword named entities required a Hungarian-specific solution. According to the UD principles, the first token of the multiword expressions should be marked as the head. However, in Hungarian, it is always the last element of the multiword expression that is inflected. Examples 15-16 demonstrate that the first element cannot be inflected, only the last one:

- (15) Találkoztam Kovács Jánossal .
meet-PAST-1SG Kovács János-INSTR .
I met János Kovács⁴.

⁴The standard order of person names is surname + first

- (16) *Találkoztam Kováccsal János .
meet-PAST-1SG Kovács-INSTR János .
I met János Kovács.

Due to the above morphosyntactic facts, we marked the last token of multiword named entities as the head in the Hungarian UD treebank while all the other UD treebanks mark the first token as the head.

4.4 Dative forms

In Hungarian, nouns that bear the suffix *-nAk* can fulfill several grammatical roles in the sentence such as:

- (17) indirect object:

Laci adott a
Leslie give-PAST-3SG the
barátjának egy almát .
friend-3SGPOSS-DAT an apple-ACC .

Leslie gave an apple to his friend.

- (18) possessor:

Laci elvette a
Leslie take-PAST-3SGOBJ the
barátjának a
friend-3SGPOSS-DAT the
könyvét .
book-3SGPOSS-ACC .

Leslie took his friend's book.

- (19) dativus ethicus:

Nekem nehogy eladd
I-DAT so.as.not.to sell-IMP-2SGOBJ
az autódat !
the car-2SGPOSS-ACC !

As for me, you should not sell you car.

- (20) experiencer:

Nekem nagyon tetszett az
I-DAT very like-PAST-3SG the
előadás .
performance .

I really liked the performance.

- (21) semantic subject:

name in Hungarian.

Lacinak bocsánatot
Leslie-DAT apology-ACC
kellett kérnie a
must-PAST-3SG ask-INF-3SG the
barátjától .
friend-3SGPOSS-ABL .

Leslie had to apologize to his friend.

While these forms do not show any difference at the morphological level, they have very different roles at the syntactic and semantic levels. Thus we decided not to make any distinction in the morphological annotation but they should have different syntactic labels. Indirect objects are marked with the label *iobj*, possessors with the label *nmod:poss* and other occurrences with *nmod:obl*. Obviously, these annotations had to be carried out manually as most of these cases could not be easily and unequivocally converted to the UD format only on the basis of morphology and syntax. Consider the following examples (Example 19 is repeated for convenience):

- (22) Nekem nehogy eladd
I-DAT so.as.not.to sell-IMP-2SGOBJ
az autódat !
the car-2SGPOSS-ACC !

As for me, you should not sell your car.

- (23) Nehogy eladd nekem az
so.as.not.to sell-IMP-2SGOBJ I-DAT the
autódat !
car-2SGPOSS-ACC !

You should not sell your car to me.

Example 22 contains a dativus ethicus whereas Example 23 contains an indirect object. The two sentences only have different word orders thus their automatic distinction would not be straightforward.

4.5 Light verb constructions

Light verb constructions are verb + noun combinations where most of the semantic content of the whole expression is carried by the noun while the syntactic head is the verb (e.g. *to have a shower*, *to make a decision*). They are not uniformly treated in Version 1.3 of the UD treebanks. Light verb constructions are either not marked at all or if they are marked, they may have a special structure or special labels (Nivre and Vincze, 2015). The Hungarian treebank belongs to the latter group, that is, members of light verb constructions bear a special

label. For instance, Figure 3 shows that the label `dobj:lvc` can be found between the nominal and verbal component of the light verb construction *döntést hoz* (decision-ACC bring) “to make a decision”. In this way, the `dobj` part of the label marks that syntactically it is a verb-object relation but semantically, it is a light verb construction, marked by the `lvc` extension of the label.

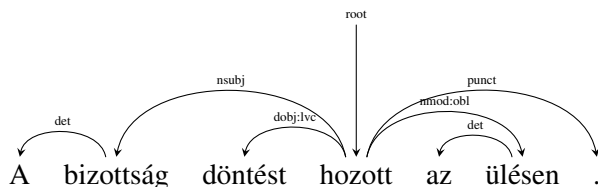


Figure 3: Light verb construction in the Hungarian UD treebank (*A bizottság döntést hozott az ülésen* (the committee decision-ACC bring-PAST-3SG the meeting-SUP) “The committee made a decision at the meeting”).

5 Experiments

We developed a converter from the existing Szeged Dependency Treebank (Vincze et al., 2010) to UD and manually corrected 1,800 sentences from the newspaper domain. The manually corrected UD sentences are available in the UD repository v3.0. The experiences gained during the manual correction could reinforce the linguistic conversion rules and the manually corrected gold standard corpus provides the opportunity for empirical evaluations which we introduce in this section.

5.1 On the Accuracy of Automatic Converters

Most of the UD treebanks are the result of automatic conversion from a dependency treebank of originally different principles. The accuracy of these automatic converters is unknown, i.e. we do not know how much information was lost or how much noise was introduced by the converters. To empirically investigate this in the case of Hungarian UD, we compared the converted and the manually corrected, i.e. gold standard, trees of the 1800 sentences.

The converter itself is based on linguistic rules (it is available on our website⁵) which were itera-

⁵<http://rgai.inf.u-szeged.hu/dependency>

tively improved by manually investigating the results of conversion on sentences of the Szeged Dependency Treebank. The final version of the converted achieves an UAS of 87.81 and a LAS of 75.99 on the 1800 sentences compared against the manually corrected UD trees. We believe that this level of accuracy is not sufficient for releasing the rest of the 80,000 sentences of the automatically converted Szeged Dependency Treebank. On the other hand, some of the shortcomings of the automatic conversion could be corrected by exploiting annotation found in other versions of the Szeged Treebank. For instance, the type of certain subordinate clauses is marked in the constituency version of the treebank, which can be transformed into UD labels. Moreover, coreference annotations from the subcorpora annotated for coreference relations could enhance the proper attachment of relative clauses. We intend to add these pieces of information to our converter in the future, hence higher accuracy scores can be provisioned for the automatic conversion process: just with the above mentioned corrections, an additional 6 percentage points could be achieved in terms of LAS as about 20% of the errors are due to subordinate or relative clauses.

5.2 On the Price of Universality

We carried out experiments for investigating whether there is any difference between using the original MSD (Vincze et al., 2014) and the new universal morphological (UM) descriptions. We were particularly interested in the utility of the two representations for dependency parsing. We trained two models of the MarMot morphological tagger (Mueller et al., 2013) using the two morphological representation in 10-fold cross-tagging on our manually corrected 1800 sentences. Then we trained and evaluated the Bohnet dependency parser (Bohnet, 2010) on the train/test split of the UD repository v3.0 utilizing the two different predicted morphological descriptions. We used the default parameters for both the MarMot and the Bohnet parser.

Table 1 presents unlabeled (UAS) and labeled (LAS) attachment scores achieved by the parser on the test set. The first column of the table indicates whether the universal morphology (UM) or the original MSD morphological codes were employed in the experiment. The second column of the table shows which dependency label set

Morph. labels	Dep labels	UAS	LAS (full label)	LAS (main label)
UM	full label	81.94	76.98	78.39
MSD	full label	82.27	77.50	78.75
UM	main label	81.70	–	78.39
MSD	main label	82.17	–	78.58

Table 1: Dependency parsing results on the Hungarian Universal Dependency dataset. In the case of *LAS(main label)* we do not check the language specific part of the dependency labels in the evaluations while we compare the universal and language-specific dependency labels at *LAS(full label)*.

was used for training the Bohnet parser. *main label* refers here to the universal dependency labels while *full label* refers to using the concatenation of universal and language-specific labels. The difference between the last two columns of the table is that we checked the full or only the main dependency labels at evaluations.

Table 1 shows the MSD outperforms UM consistently at each of the experiments. Although these differences are not high, this suggests that some information encoded in the MSD morphology is not represented in UM, i.e. we have to pay a price to be universal. We can observe the greatest difference when training and evaluating on full dependency labels, i.e. language-specific morphological features contribute to the prediction of language-specific dependency labels.

We made a manual error analysis of the results with regard to attachment (UAS) errors, i.e. we compared the outputs of the dependency parsers trained by using predicted universal codes and predicted MSD morphological codes, respectively. Results are presented in Table 2. We found that the benefits of the original language-specific annotation (MSD) mostly manifests in the treatment of subordinate clauses, adverbial modifiers and infinitival complements. These results might be explained by the fact that in certain cases, MSD contains more detailed grammatical information than the UM formalism. For instance, MSD encodes whether a conjunction connects clauses or words/phrases, which information is missing from UM. Also, higher results were achieved for cases when two nouns or adjectives were following each other and one of them modified the other (as in *magas rangú képviselői* “representatives of high standings”). However, sentences containing an overt or covert form of the copula could be parsed more effectively by using universal morphology codes.

Error type	MSD	%	UM	%
Coordination	100	32.05	98	30.34
Article	44	14.10	44	13.62
Adverbial	35	11.22	43	13.31
Other	37	11.86	31	9.60
Part/adj compl.	31	9.94	32	9.91
Adjacent N/A	15	4.81	20	6.19
Subordination	13	4.17	17	5.26
Copula	14	4.49	11	3.41
Infinitive	9	2.88	15	4.64
Nominal arg.	8	2.56	8	2.48
Possessor	6	1.92	4	1.24
Total	312	100	323	100

Table 2: Error analysis: the number and ratio of specific error types.

5.3 The Added Value of Language-specific UD Labels

We also investigated the impact of the language-specific parts of the dependency labels. As the numbers in Table 1 show, slightly better results can be achieved both in terms of UAS and LAS when training the model with *full labels* than with *main labels*. This highlights the importance of adding language specific distinctions to the universal ones because they may contain information that can be exploited during the tree decoding. They contribute even to unlabeled attachment decisions. To take an example, UD does not make any distinction among different types of nominal modifiers, treating them as `nmod`. However, for Hungarian, we applied extra labels such as `nmod:poss` for possessors (see Section 3.1) and `nmod:obl` for nominal arguments of the verb. As for the first, it should always be attached to the possessed noun, whereas the second one is attached to a verb (see also Examples 18 and 19 with the dative morphological case). Thus, the parser can learn these fine-grained distinctions, which might be beneficial for the unlabeled attachment scores as well.

Also, we would like to point out that the utilization of language-specific labels does not contradict the UD principles. In UD, each language should select the appropriate labels according to their needs but there is no need to apply all of the labels/features. General labels like `nsubj` or `dobj` will be used in most (maybe all) of the UD languages but there are other labels or feature-value pairs that are applicable for only a handful of languages. These ones are now called as “language-specific” features but in principle, their status is not different from those that are more widely applied. So we believe that introducing “language-specific” additions does not harm the UD principles. Moreover, the chief objective of our experiments was to highlight the added value of language-specific features and we were able to show that they can even improve parsing accuracy when evaluated exclusively on the general labels. The main goal of UD is to provide a way where the parsing results over languages are comparable, hence using language specific features during decoding but evaluating only on general labels is in line with this comparison principle. Moreover, it indicates for UD treebank developers that – besides general labels – language-specific ones have to be taken seriously.

6 Conclusions

In this paper, the principles of universal dependencies and morphology for Hungarian were introduced by reporting the most challenging grammatical phenomena and our solutions to those. We converted then manually corrected 1,800 sentences from the Szeged Treebank to universal dependency format and introduced experiments on this manually annotated corpus for evaluating automatic conversion and the added value of language-specific, i.e. non-universal, annotations. We would like to draw the attention to the importance of understanding i) the information loss of the automatic UD converters; ii) what is the price of being constrained by universal morphology principles and; iii) the utility of exploiting language-specific dependency labels in UD.

Acknowledgments

The research of Richárd Farkas was funded by the János Bolyai Scholarship.

References

- Maria Jesus Aranzabe, Aitziber Atutxa, Kepa Ben-goetxea, Arantza Diaz de Illaraza, Iakes Goenaga, Koldo Gojenola, and Larraitz Uriá. 2014. Automatic Conversion of the Basque Dependency Treebank to Universal Dependencies. In *Proceedings of the Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*.
- Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97.
- Dóra Csendes, János Csirik, and Tibor Gyimóthy. 2004. The Szeged Corpus. A POS Tagged and Syntactically Annotated Hungarian Natural Language Corpus. In *COLING 2004 5th International Workshop on Linguistically Interpreted Corpora*, pages 19–22.
- Kaja Dobrovoljc and Joakim Nivre. 2016. The Universal Dependencies Treebank of Spoken Slovenian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Tomaž Erjavec. 2012. MULTTEXT-East: morphosyntactic resources for Central and Eastern European languages. *Language Resources and Evaluation*, 46(1):131–142.
- Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze. 2013. Knowledge sources for constituent parsing of german, a morphologically rich and less-configurational language. *Computational Linguistics*, 39(1):57–85.
- Katri Haverinen, Jenna Nyblom, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Stina Ojala, Tapio Salakoski, and Filip Ginter. 2014. Building the essential resources for finnish: the turku dependency treebank. *Language Resources and Evaluation*, 48(3):493–531.
- Anders Johannsen, Héctor Martínez Alonso, and Barbara Plank. 2014. Universal Dependencies for Danish. In *Proceedings of the Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.
- Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332.

- Kadri Muischnek, Kaili Müürisep, and Tiina Puolakainen. 2016. Estonian Dependency Treebank: from Constraint Grammar tagset to Universal Dependencies. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Joakim Nivre and Veronika Vincze. 2015. Light verb constructions in universal dependencies. Poster at the 5th PARSEME meeting, Iași, Romania.
- Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Joakim Nivre. 2014. Universal Dependencies for Swedish. In *Proceedings of SLTC 2014*.
- Joakim Nivre. 2015. Towards a Universal Grammar for Natural Language Processing. In *Computational Linguistics and Intelligent Text Processing*, pages 3–16.
- Lilja Øvrelid and Petter Hohle. 2016. Universal Dependencies for Norwegian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC*.
- Alain Polguère and Igor Aleksandrovič Mel’čuk, editors. 2009. *Dependency in Linguistic Description*. Studies in language companion series.
- Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal Dependencies for Finnish. In *Proceedings of Nodalida*.
- Owen Rambow, Bonnie Dorr, David Farwell, Rebecca Green, Nizar Habash, Stephen Helmreich, Eduard Hovy, Lori Levin, Keith J. Miller, Teruko Mitamura, Reeder, Florence, and Advait Siddharthan. 2006. Parallel syntactic annotation of multiple languages. In *Proceedings of LREC*.
- Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galleitebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Yuval Marton, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, and Alina Wróblewska. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182.
- Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109.
- Mojgan Seraji, Filip Ginter, and Joakim Nivre. 2016. Universal Dependencies for Persian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Takaaki Tanaka, Yusuke Miyao, Masayuki Asahara, Sumire Uematsu, Hiroshi Kanayama, Shinsuke Mori, and Yuji Matsumoto. 2016. Universal Dependencies for Japanese. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Miklós Törkenczy. 2005. *Practical Hungarian Grammar*. Corvina, Budapest.
- Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. In *Proceedings of LREC 2010*.
- Veronika Vincze, Viktor Varga, Katalin Ilona Simkó, János Zsibrita, Ágoston Nagy, Richárd Farkas, and János Csirik. 2014. Szeged Corpus 2.5: Morphological Modifications in a Manually POS-tagged Hungarian Corpus. In *Proceedings of LREC 2014*, pages 1074–1078.
- Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*.
- Daniel Zeman. 2015. Slavic Languages in Universal Dependencies. In *Slovko 2015: Natural Language Processing, Corpus Linguistics, E-learning*.

Addressing the Data Sparsity Issue in Neural AMR Parsing

Xiaochang Peng^{*1}, Chuan Wang^{*2}, Daniel Gildea¹ and Nianwen Xue²

¹University of Rochester
{xpeng, gildea}@cs.rochester.edu

²Brandeis University
{cwang24, xuen}@brandeis.edu

Abstract

Neural attention models have achieved great success in different NLP tasks. However, they have not fulfilled their promise on the AMR parsing task due to the data sparsity issue. In this paper, we describe a sequence-to-sequence model for AMR parsing and present different ways to tackle the data sparsity problem. We show that our methods achieve significant improvement over a baseline neural attention model and our results are also competitive against state-of-the-art systems that do not use extra linguistic resources.

1 Introduction

Abstract Meaning Representation (AMR) (Banasescu et al., 2013) is a semantic formalism where the meaning of a sentence is encoded as a rooted, directed graph. Figure 1 shows an example of an AMR in which the nodes represent the AMR concepts and the edges represent the relations between the concepts they connect. AMR concepts consist of predicate senses, named entity annotations, and in some cases, simply lemmas of English words. AMR relations consist of core semantic roles drawn from the Propbank (Palmer et al., 2005) as well as very fine-grained semantic relations defined specifically for AMR. These properties render the AMR representation useful in applications like question answering and semantics-based machine translation.

The task of AMR graph parsing is to map natural language strings to AMR semantic graphs. Recently, a sizable new corpus of English/AMR pairs (LDC2015E86) has been released. Different parsers have been developed to tackle this problem (Flanigan et al., 2014; Wang et al., 2015b;

^{*}Both authors contribute equally.

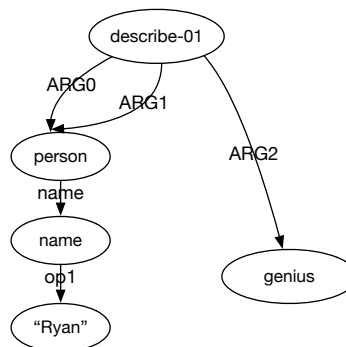


Figure 1: An example of AMR graph representing the meaning of: “Ryan’s description of himself: a genius.”

Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015). Most of these parsers have used external resources such as dependency parses, semantic lexicons, etc., to tackle the sparsity issue.

Recently, Sutskever et al. (2014) introduced a neural network model for solving the general sequence-to-sequence problem, and Bahdanau et al. (2014) proposed a related model with an attention mechanism that is capable of handling long sequences. Both models achieve state-of-the-art results on large scale machine translation tasks.

However, sequence-to-sequence models mostly work well for large scale parallel data, usually involving millions of sentence pairs. Vinyals et al. (2015) present a method which linearizes parse trees into a sequence structure and therefore a sequence-to-sequence model can be applied to the constituent parsing task. Competitive results have been achieved with an attention model on the Penn Treebank dataset, with only 40K annotated sentences.

AMR parsing is a much harder task in that the target vocabulary size is much larger, while the size of the dataset is much smaller. While for constituent parsing we only need to predict non-

terminal labels and the output vocabulary is limited to 128 symbols, AMR parsing has both concepts and relation labels, and the target vocabulary size consists of tens of thousands of symbols. Barzdins and Gosko (2016) applied a similar approach where AMR graphs are linearized using depth-first search and both concepts and relations are treated as tokens (see Figure 3). Due to the data sparsity issue, their AMR parsing results are significantly lower than state-of-the-art models when using the neural attention model.

In this paper, we present a method which linearizes AMR graphs in a way that captures the interaction of concepts and relations. To overcome the data sparsity issue for the target vocabulary, we propose a categorization strategy which first maps low frequency concepts and entity subgraphs to a reduced set of category types. In order to map each type to its corresponding target side concepts, we use heuristic alignments to connect source side spans and target side concepts or subgraphs. During decoding, we use the mapping dictionary learned from the training data or heuristic rules for certain types to map the target types to their corresponding translation as a postprocessing procedure.

Experiments show that our linearization strategy and categorization method are effective for the AMR parsing task. Our model improves significantly in comparison with the previously reported sequence-to-sequence results and provides a competitive benchmark in comparison with state-of-the-art results without using dependency parses or other external semantic resources.

2 Sequence-to-sequence Parsing Model

Our model is based on an existing sequence-to-sequence parsing model (Vinyals et al., 2015), which is similar to models used in neural machine translation.

2.1 Encoder-Decoder

Encoder. The encoder learns a context-aware representation for each position of the input sequence by mapping the inputs w_1, \dots, w_m into a sequence of hidden layers h_1, \dots, h_m . To model the left and right contexts of each input position, we use a bidirectional LSTM (Bahdanau et al., 2014). First, each input’s word embedding representation x_1, \dots, x_m is obtained through a lookup table. Then these embeddings serve as the input to

two RNNs: a forward RNN and a backward RNN. The forward RNN can be seen as a recurrent function defined as follows:

$$h_i^{fw} = f(x_i, h_{i-1}^{fw}) \quad (1)$$

Here the recurrent function f we use is Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber, 1997). The backward RNN works similarly by repeating the process in reverse order. The outputs of forward RNN and backward RNN are then depth-concatenated to get the final representation of the input sequence.

$$h_i = [h_i^{fw}, h_{m-i+1}^{bw}] \quad (2)$$

Decoder. The decoder is also an LSTM model which generates the hidden layers recurrently. Additionally, it utilizes an attention mechanism to put a “focus” on the input sequence. At each output time step j , the attention vector d'_j is defined as a weighted sum of the input hidden layers, where the masking weight α_i^j is calculated using a feed-forward neural network. Formally, the attention vector is defined as follows:

$$u_i^j = v^T \tanh(W_1 h_i + W_2 d_j) \quad (3)$$

$$\alpha_i^j = \text{softmax}(u_i^j) \quad (4)$$

$$d'_j = \sum_{i=1}^m \alpha_i^j h_i \quad (5)$$

where d_j is the output hidden layer at time step j , and v , W_1 , and W_2 are parameters for the model. Here the weight vector $\alpha_1^j, \dots, \alpha_m^j$ is also interpreted as a soft alignment in the neural machine translation model, which similarly could also be treated as a soft alignment between token sequences and AMR relation/concept sequences in the AMR parsing task. Finally, we concatenate the hidden layer d_j and attention vector d'_j to get the new hidden layer, which is used to predict the output sequence label.

$$P(y_j | w, y_{1:j-1}) = \text{softmax}(W_3 [d_j, d'_j]) \quad (6)$$

2.2 Parse Tree as Target Sequence

Vinyals et al. (2015) designed a reversible way of converting the parse tree into a sequence, which they call linearization. The linearization is performed in the depth-first traversal order. Figure 2 shows an example of the linearization result. The target vocabulary consists of 128 symbols.

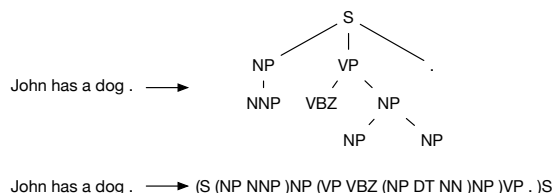


Figure 2: Example parsing task and its linearization.

In practice, they found that using the attention model is more data efficient and works well on the parsing task. They also reversed the input sentence and normalized the part-of-speech tags. After decoding, the output parse tree is recovered from the output sequence of the decoder in a post-processing procedure. Overall, the sequence-to-sequence model is able to match the performance of the Berkeley Parser (Petrov et al., 2006).

3 AMR Linearization

Barzdins and Gosko (2016) present a similar linearization procedure where the depth-first traversal result of an AMR graph is used as the AMR sequence (see Figure 3). The bracketing structure of AMR is hard to maintain because the prediction of relation (with left parenthesis) and the prediction of an isolated right parenthesis are not correlated. As a result, the output AMR sequences usually have parentheses that do not match.

We present a linearization strategy which captures the bracketing structure of AMR and the connection between relations and concepts. Figure 3 shows the linearization result of the AMR graph shown in Figure 1. Each relation connects the head concept to a subgraph structure rooted at the tail concept, which shows one branch below the head concept. We use the relation label and left parenthesis to show the beginning of the branch (subgraph) and use right parenthesis paired with the relation label to show the end of the branch. We additionally add “-TOP-” at the beginning to show the start of the traversal of the AMR graph and add “)-TOP-” at the end to show the end of traversal. When a symbol is revisited, we replace the symbol with “-RET-”. We additionally add the revisited symbol before “-RET-” to decide where the reentrancy is introduced to.¹ We also get rid of

¹This is an approximation because one concept can appear multiple times, and we simply attach the reentrancy to the most recent appearance of the concept. An additional index would be needed to identify the accurate place of reentrancy.

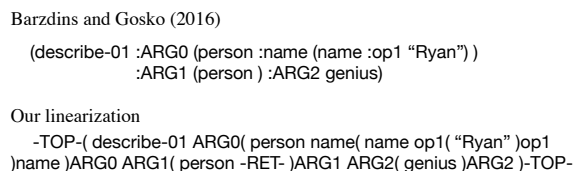


Figure 3: Comparison of AMR linearization

variables and only keep the full concept label. For example, “g / genius” to “genius”.

We can easily recover the original AMR graph from its linearized sequence. The sequence also captures the branching information of each relation explicitly by representing it with a start symbol and an end symbol specific to that relation. During our experiments, most of the output sequences have a matching bracketing structure using this linearization strategy. The idea of linearization is basically a depth-first traversal of the AMR where the original graph structure can be reconstructed with the linearization result. Even though we call it a sequence, its core idea is actually generating a graph structure from top-down.

4 Dealing with the Data Sparsity Issue

While sequence-to-sequence models can be successfully applied to constituent parsing, they do not work well on the AMR parsing task as shown by Barzdins and Gosko (2016). The main bottleneck is that the size of target vocabulary for AMR parsing is much larger than constituent parsing, tens of thousands in comparison with 128, and the size of training data is less than half of that available for parsing.

In this section, we present a categorization method which significantly reduces the target vocabulary size, as the alignment from the attention model does not work well on the relatively small dataset. To adjust for the alignment errors made by the attention model, we propose to add supervision from an alignment produced by an external aligner which can use lexical information to overcome the limit of data size.

4.1 AMR Categorization

We define several types of categories and map low frequency words into these categories.

1. DATE: we reduce all the date entity subgraphs to this category, ignoring details of the specific date entity.

Sentence side (lemmatized, lower cased)	AMR side
Before linearization:	
chinese seismology be gallop down the wrong road .	(g / gallop-01 :ARG0 (s / seismology :mod (c / country :wiki "China" :name (n / name :op1 "China"))) :ARG1 (r / road :mod (w / wrong)) :direction (d / down))
After linearization:	
NE_country-0 -SURF--0 be -VERB--0 down the wrong -SURF-1 .	-TOP-(-VERB--0 ARG0(-SURF--0 mod(NE_country-0)mod)ARG0 ARG1(-SURF-1 mod(wrong)mod)ARG1 direction(down)direction)-TOP-

Figure 4: An example of categorized sentence-AMR pair.

2. NE- $\{ent\}$: we reduce all named entity subgraphs to this category, where *ent* is the root label of each subgraph, such as *country* or *person*.
3. -VERB-: we map predicate variables with low frequency ($n < 50$) to this category
4. -SURF-: we map non-predicate variables with low frequency ($n < 50$) to this category
5. -CONST-: we map constants other than numbers, “-”, “interrogative”, “expressive”, “imperative” to this category.
6. -RET-: we map all revisited concepts to this category.
7. -VERBAL-: we additionally use the verbalization list ² from the AMR website and map matched subgraphs to this category.

After the re-categorization, the vocabulary size is substantially reduced to around 2000, though this vocabulary size is still very large for the relatively small dataset. These categories and the frequent concepts amount to more than 90% of all the target words, and each of these are learned with a larger number of occurrences.

4.2 Categorize Source Sequence

The source side tokens also have sparsity issues. For example, even if we have mapped the number *1997* to “DATE”, we can not easily generalize it

²<http://amr.isi.edu/download/lists/verbalization-list-v1.06.txt>

to the token *1993* if it does not appear in the training data. Also, some special 6-digit date formats such as “YYMMDD” are hard to learn using co-occurrence statistics.

Our basic approach to dealing with this issue is to generalize these sparse tokens or spans to some special categories (currently we use the same set of categories defined in the previous section). On the training data, we can use the heuristic alignment. For example, if we learned from the heuristic alignment that “010911” is aligned to a date-entity of September 11, 2001 on the AMR side, we use the same category “DATE” to replace this token. We distinguish this alignment from other date alignments by assigning a unique indexed category “DATE-X” to both sides of the alignment, where “X” counts from 0 and adds one for each new date entity from left to right on the sentence side. The same index strategy goes for all the other categories. Figure 4 shows an example of the linearized parallel sequence. The first infrequent non-predicate variable “seismology” is mapped to “-SURF-0”, then “wrong” to “-SURF-1” based on its position on the sentence side. The indexed category labels are then projected onto the target side based on the heuristic alignment. During this re-categorization procedure, we build a map Q from each token or span to its most likely concept or category on the target side based on relative frequency. We also dump a DATE template for recognizing new date entities by abstracting away specific date fields such as “1997” to “YEAR”, “September” to “MONTH”. For example, we build a template “MONTH DAY, YEAR”

from the specific date mention “June 6, 2007”.

During decoding, we are only given the sentence. We first use the date templates learned from the training data to recognize dates in each sentence. We also use a named entity tagger to recognize named entity mentions in the sentence. We use the entity name and wiki information from Q if there is a match of the entity mention, otherwise for convenience we simply use “person” as the entity name and use wiki “-”. For each of the other tokens, we first look it up in Q and replace it with the most likely mapping. If there is no match, we further look it up in the verbalization list. In case there is still no match, we use the part of speech information to assign its category. We replace verbs with category “-VERB-” and nouns with category “-SURF-”. In accordance with the categorized token sequence, we also index each category in the resulting sequence from left to right.

4.3 Recovering AMR graph

During decoding, our output sequences usually have categories and we need to map each category to AMR concepts or subgraphs. When we categorize the tokens in each sentence before decoding, we save the mapping from each category to its original token as table D . As we use the same set of categories on both source and target sides, we heuristically align target side category label to its source side counterpart from left to right. Given table D , we know which source side token it comes from and use the most frequent concept or subgraph of the token to replace the category.

4.4 Supervised Attention Model

In this section, we propose to learn the attention vector in a supervised manner. There are two main motivations behind this. First, the neural attention model usually utilizes millions of data points to train the model, which learns a quite reasonable attention vector that at each output time step constrains the decoder to put a focus on the input sequences (Bahdanau et al., 2014; Vinyals et al., 2015). However, we only have 16k sentences in the AMR training data and our output vocabulary size is quite large, which makes it hard for the model to learn a useful alignment between the input sequence and AMR concepts/relations. Second, as argued by Liu et al. (2016), the sequence-to-sequence model tries to calculate the attention vector and predict the current output label simultaneously. This makes it impossible for the learned

soft alignment to combine information from the whole output sentence context. However, traditional word alignment can easily use the whole output sequence, which will produce a more informed alignment.

Similar to the method used by Liu et al. (2016), we add an additional loss to the original objective function to model the disagreement between the reference alignment and the soft alignment produced by the attention mechanism. Formally, for each input/output sequence pair (\mathbf{w}, \mathbf{y}) in the training set, the objective function is defined as:

$$-\frac{1}{n} \sum_{j=1}^n \log p(y_j | \mathbf{w}, y_{1:j-1}) + \lambda \Theta(\bar{\alpha}^j, \alpha^j) \quad (7)$$

where $\bar{\alpha}^j$ is the reference alignment for output position j , which is provided by the existing aligner, α^j is the soft alignment, $\Theta()$ is cross-entropy function, n is the length of output sequence and λ is the hyperparameter which serves as a trade-off between sequence prediction and alignment supervision. Note that the supervised attention part doesn’t affect the decoder which will predict the output label given learned weights.

One issue with this method is how we represent $\bar{\alpha}$. As the output of conventional aligner is a hard decision, alignment is either one or zero for each input position. In addition, multiple input words could be aligned to one single concept. Finally, in the AMR sequences, there are many output labels (mostly relations) which don’t align to any word in the input sentence. We utilize a heuristic method to process the reference alignment. We assign an equal probability among the words that are aligned to one AMR concept. Then if the output label doesn’t align to any input word, we assign an even probability for every input word.

5 Experiments

We evaluate our system on the released dataset (LDC2015E86) for SemEval 2016 task 8 on meaning representation parsing (May, 2016). The dataset contains 16,833 training, 1,368 development and 1,371 test sentences which mainly cover domains like newswire, discussion forum, etc. All parsing results are measured by Smatch (version 2.0.2) (Cai and Knight, 2013).

5.1 Experiment Settings

We first preprocess the input sentences with tokenization and lemmatization. Then we extract

the named entities using the Illinois Named Entity Tagger (Ratinov and Roth, 2009).

For training all the neural AMR parsing systems, we use 256 for both hidden layer size and word embedding size. Stochastic gradient descent is used to optimize the cross-entropy loss function and we set the drop out rate to be 0.5. We train our model for 150 epochs with initial learning rate of 0.5 and learning rate decay factor 0.95 if the model doesn't improve for the 3 last epochs.

5.2 Baseline Model

Our baseline model is a plain sequence-to-sequence model which has been used in the constituent parsing task (Vinyals et al., 2015). While they use a 3-layer deep LSTM, we only use a single-layer LSTM for both encoder and decoder since our data is relatively small and empirical comparison shows that stacking more layers doesn't help in our case. AMR linearization follows Section 3 without categorization. Since we don't restrict the input/output vocabulary in this case, our vocabulary size is quite large: 10,886 for output vocabulary and 2,2892 for input vocabulary. We set them to 10,000 and 20,000 respectively and replace the out of vocabulary words with `_UNK_`.

5.3 Impact of Re-Categorization

We first inspect the influence of utilizing categorization on the input and output sequence. Table 1 shows the Smatch evaluation score on development set.

System	P	R	F
Baseline	0.42	0.34	0.37
Re-Categorization ($n = 50$)	0.55	0.46	0.50

Table 1: Re-Categorization impact on development set

We see from the table that re-categorization improves the F-score by 13 points on the development set. As mentioned in section 4.1, by setting the low frequency threshold n to 50 and re-categorizing them into a reduced set of types, we now reduce the input/output vocabulary size to (2,000, 6,000). This greatly reduces the label sparsity and enables the neural attention model to learn a better representation on this small scale data. Another advantage of this method

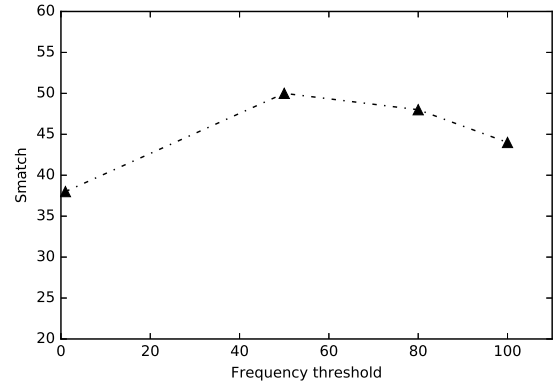


Figure 5: AMR parsing performance on development set given different categorization frequency.

is that although AMR tries to abstract away surface forms and retain the semantic meaning structure of the sentence, a large portion of the concepts are coming from the surface form and have exactly same string form both in input sentence and AMR graph. For example, `nation` in sentence is mapped to concept (`n / nation`) in the AMR. For the frequent concepts in the output sequence, since the model can observe many training instances, we assume that it can be predicted by the attention model. For the infrequent concepts, because of the categorization step, we only require the model to predict the concept type and its relative position in the graph. By applying the post-processing step mentioned in Section 4.3, we can easily recover the categorized concepts to their original form.

We also inspect how the value of re-categorization frequency threshold n affects the AMR parsing result. As shown in Figure 5, setting n to 0, which means no output labels will be categorized into types `-VERB-` and `-SURF-`, doesn't improve the baseline system. The reason is that we still have a large output vocabulary size and training data is still sparse with respect to the low frequency output labels. Also, if we set n too high, although the output vocabulary size becomes smaller, some of the frequent output labels that the model handles well originally will be put into the coarse-grained types, losing information in the recovery process. Thus we can see from the plot that after the optimal point the Smatch score will drop. Therefore, we choose to set $n = 50$ in the subsequent experiments.

5.4 Impact of Supervised Alignment

Choice of External Aligner. There are two existing AMR aligners: one is a rule-based aligner coming with JAMR (Flanigan et al., 2014), which defines regular expression patterns to greedily match between AMR graph fragment and input token spans; another one is an unsupervised aligner (Pourdamghani et al., 2014) which adopts the traditional word alignment method in machine translation. Although evaluated on different set of manual alignment test sentences, both aligners achieved $\sim 90\%$ F1 score. Here we choose to use the second aligner, as it covers broader domains.

Different alignment configurations To balance between the sequence learning and alignment agreement, We empirically tune the hyperparameter λ and set it to 0.3. For the external alignment we use for reference, we convert it to a vector with equal probability as discussed in Section 4.4. We then train a sequence-to-sequence model with re-categorized input/output and report the result on development set.

System	P	R	F
Baseline	0.42	0.34	0.37
Categorization ($n = 50$)	0.55	0.46	0.50
SuperAttn+Cate ($n = 50$)	0.56	0.49	0.52

Table 2: Supervised attention impact on development set

As shown in Table 2, the supervised attention model is able to further improve the Smatch score by another 2 points, which are mainly contributed by 3 points increase in recall. Since the reference/external alignment is mostly between the input tokens and AMR graph concepts, we believe that the supervised attention model is able to constrain the decoder so that it will output concepts which can be aligned to some tokens in the input sentence.

System	P	R	F
SuperAttn+Cate ($n = 50$)	0.56	0.49	0.52
NO-RELATION-ALIGN	0.46	0.40	0.43

Table 3: Supervised attention impact on development set

Because we have relations in the AMR graph, the alignment problem here is different from the

word alignment in machine translation. To verify the effectiveness of our setup, we also compare our configuration to the condition **NO-RELATION-ALIGN** where we only ignore the alignment between sentence and AMR relations by putting an all zero vector as the reference attention for each output relation label. From Table 3 we see that simply ignoring the reference attention for relations would greatly affect the model performance, and how we effectively represent the reference alignment for relations is crucial for the supervised attention model.

5.5 Results

In this section we report our final result on the test set of SemEval 2016 Task 8 and compare our model with other parsers. We train our model utilizing re-categorization and supervised attention with hyperparameters tuned on the development set. Then we apply our trained model on the test set.

Firstly, we compare our model to the existing sequence-to-sequence AMR parsing model of Barzdins and Gosko (2016). As shown in table 4, the word-level model in Barzdins and Gosko (2016) is basically our baseline model. The second model they use is a character-based sequence-to-sequence model. Our model can also be regarded as a word-level model; however, by utilizing carefully designed categorization and supervised attention, our system outperforms both their results by a large margin.

System	P	R	F
Our system	0.55	0.50	0.52
Barzdins and Gosko (2016) [†]	-	-	0.37
Barzdins and Gosko (2016) [*]	-	-	0.43

Table 4: Compare to other sequence-to-sequence AMR parser. Barzdins and Gosko (2016)[†] is the word-level neural AMR parser, Barzdins and Gosko (2016)^{*} is the character-level neural AMR parser.

Table 5 gives the comparison of our system to some of the teams participating in SemEval16 Task 8. Since a large portion of the teams extend on the state-of-the-art system CAMR (Wang et al., 2015b; Wang et al., 2015a; Wang et al., 2016), here we just pick typical teams that represent different approaches. We can see from the table that our system fails to outperform the state-

of-the-art system. However, the best performing system CAMR uses a dependency structure as a starting point, where dependency parsing has achieved high accuracy recently and can be trained on larger corpora. Also, it utilizes semantic role labeling and complex features, which makes the training process a long pipeline. Our system only needs minimal preprocessing, and doesn't need the dependency parsing step. Our approach is competitive with the SHRG (Synchronous Hyperedge Replacement Grammar) method of Peng et al. (2015), which does not require a dependency parser and uses SHRG to formalize the string-to-graph problem as a chart parsing task. However, they still need a concept identification stage, while our model can learn the concepts and relations jointly.

System	P	R	F
Our system	0.55	0.50	0.52
Peng and Gildea (2016)	0.56	0.55	0.55
CAMR	0.70	0.63	0.66

Table 5: Comparison to other AMR parsers.

6 Discussion

In this paper, we have proposed several methods to make the sequence-to-sequence model work competitively against conventional AMR parsing systems. Although we haven't outperformed state-of-the-art system using the conventional methods, our results show the effectiveness of our approaches to reduce the sparsity problem when training sequence-to-sequence model on a relatively small dataset. Our work could be aligned with the effort to handle low-resource data problems when building the end-to-end neural network model.

In neural machine translation, the attention model is traditionally trained on millions of sentence pairs, while facing low-resource language pairs, the neural MT system performance tends to downgrade (Zoph et al., 2016). There has been growing interest in tackling sparsity/low-resource problem in neural MT. Zoph et al. (2016) use a transfer learning method to first pre-train the neural model on rich-resource language pairs and then import the learned parameters to continue training on low-resource language pairs so that the model can alleviate the sparsity problem through shared

parameters. Firat et al. (2016) builds a multilingual neural system where the attention mechanism can be shared between different language pairs. Our work could be seen as parallel efforts to handle the sparsity problem since we focus on the input/output categorization and external alignment, which are both handy for low-resource languages.

In this paper, we haven't used any syntactic parser. However, as shown in previous works (Flanigan et al., 2014; Wang et al., 2015b; Artzi et al., 2015; Pust et al., 2015), using dependency features helps improve the parsing performance significantly because of the linguistic similarity between the dependency tree and AMR structure. An interesting extension would be to use a linearized dependency tree as the source sequence and apply sequence-to-sequence to generate the AMR graph. Our parser could also benefit from the modeling techniques in Wu et al. (2016).

7 Conclusion

Neural attention models have achieved great success in different NLP tasks. However, they have not been as successful on AMR parsing due to the data sparsity issue. In this paper, we described a sequence-to-sequence model for AMR parsing and present different ways to tackle the data sparsity problems. We show that our methods have led to significant improvement over a baseline neural attention model, and our model is also competitive against models that do not use extra linguistic resources.

Acknowledgments Funded in part by a Google Faculty Award.

References

- Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation

- for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.
- Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *arXiv preprint arXiv:1604.01278*.
- Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, June. Association for Computational Linguistics.
- Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- L. Liu, M. Utiyama, A. Finch, and E. Sumita. 2016. Neural Machine Translation with Supervised Attention. *ArXiv e-prints*, September.
- Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California, June. Association for Computational Linguistics.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- Xiaochang Peng and Daniel Gildea. 2016. UofR at semeval-2016 task 8: Learning synchronous hyperedge replacement grammar for AMR parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1185–1189, San Diego, California, June. Association for Computational Linguistics.
- Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 32–41, Beijing, China, July. Association for Computational Linguistics.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.
- Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429, Doha, Qatar, October. Association for Computational Linguistics.
- Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154, Lisbon, Portugal, September. Association for Computational Linguistics.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.
- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.
- Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862, Beijing, China, July. Association for Computational Linguistics.
- Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado, May–June. Association for Computational Linguistics.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at semeval-2016 task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178, San Diego, California, June. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

B. Zoph, D. Yuret, J. May, and K. Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. *ArXiv e-prints*, April.

Generating Natural Language Question-Answer Pairs from a Knowledge Graph Using a RNN Based Question Generation Model

Sathish Indurthi [‡], Dinesh Raghu², Mitesh M. Khapra [‡] and Sachindra Joshi²

¹Samsung Electronics, DMC R&D

²IBM Research India

³Indian Institute of Technology Madras

s.indurthi@samsung.com

{diraghul, jsachind}@in.ibm.com

miteshk@cse.iitm.ac.in

Abstract

In recent years, knowledge graphs such as Freebase that capture facts about entities and relationships between them have been used actively for answering factoid questions. In this paper, we explore the problem of automatically generating question answer pairs from a given knowledge graph. The generated question answer (QA) pairs can be used in several downstream applications. For example, they could be used for training better QA systems. To generate such QA pairs, we first extract a set of keywords from entities and relationships expressed in a triple stored in the knowledge graph. From each such set, we use a subset of keywords to generate a natural language question that has a unique answer. We treat this subset of keywords as a sequence and propose a sequence to sequence model using RNN to generate a natural language question from it. Our RNN based model generates QA pairs with an accuracy of 33.61 percent and performs 110.47 percent (relative) better than a state-of-the-art template based method for generating natural language question from keywords. We also do an extrinsic evaluation by using the generated QA pairs to train a QA system and observe that the F1-score of the QA system improves by 5.5 percent (relative) when using automatically generated QA pairs in addition to manually generated QA pairs available for training.

[‡]This work was done while the author was a part of IBM Research India

[‡]This work was done while the author was a part of IBM Research India

1 Introduction

Knowledge graphs store information about millions of *things* (or entities) and relationships between them. Freebase¹ is one such knowledge graph that describes and organizes more than 3 billion *facts* in a consistent ontology. Knowledge graphs usually capture relationships between different *things* that can be viewed as triples (for example, CEO(Sundar Pichai, Google)). Such triples are often referred to as facts and can be used for answering factoid questions. For example, the above triple can be used to answer the question “*Who is the CEO of Google ?*”. It is not surprising that knowledge graphs are increasingly used for building Question Answering systems (Ferrucci, 2012; Yahya et al., 2013; He et al., 2014; Zou et al., 2014).

In this paper, we focus on exploiting knowledge graphs for a related but different purpose. We propose that such triples or facts can be used for automatically generating Question Answer (QA) pairs. The generated QA pairs can then be used in certain downstream applications. For example, if some domain-specific knowledge graphs are available (such as History, Geography) then such QA pairs generated from them can be used for developing quiz systems for educational purposes.

We now formally define the problem and then illustrate it with the help of an example. Consider a triple consisting of a subject, predicate and object. Typically, the predicate has a domain (subject type) and a range (object type) associated with it. The predicate may have zero or more parents in the knowledge graph. For the sake of simplicity let us assume that the predicate has a single parent. We define a set consisting of the subject, predicate, object, domain, range and predicate parent. We propose an approach

¹<https://www.freebase.com/>

<i>Predicate</i>	CEO
<i>Subject</i>	Sundar Pichai
<i>Object</i>	Google
<i>Parent Predicate</i>	designation
<i>Domain</i>	person
<i>Range</i>	organization
<i>Keywords</i>	CEO, designation, Sundar Pichai, person, Google, organization

Table 1: An example set of keywords constructed from the triple *CEO(Sundar Pichai, Google)*

to generate natural language factoid questions using a subset of this set such that the answer to the question also lies in the set. Given the set of keywords, as shown in Table 1, we could generate the following QA pairs (keywords are italicized):

Q: What is the *designation* of *Sundar Pichai* at *Google*?

A: *CEO*

Q: Which *organization* is *Sundar Pichai* the *CEO* of?

A: *Google*

The above problem is similar to the problem of generating questions from Web queries (instead of entities and relations) which was first suggested by Lin (Lin, 2008). However, unlike existing works on query-to-questions which mainly rely on template based approaches, we formulate this as a sequence to sequence generation problem wherein the ordered set of keywords is an input sequence and the natural language question is the output sequence. We use a Recurrent Neural Network (RNN) (Werbos, 1990; Rumelhart et al., 1988) based model with Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units to generate questions from the given set of keywords.

The input to our question generation model is a set of keywords extracted from triples in a knowledge graph. For this, it is important to first select a subset of triples from which interesting and meaningful questions can be constructed. For example, no interesting questions can be constructed from the triple *wikipedia_page_ID(Google, 57570)* and hence we should eliminate such triples. Further, even for an interesting triple, it may be possible to use only certain subsets of keywords to construct

a meaningful question. For example, for the set of keywords shown in Table 1, it is not possible to use the subset $\{person, designation\}$ to form an interesting question. Hence, we need to automatically identify the right set of keywords that should be used to form the question such that the answer also lies in the set. In addition to the question generation model, we also propose a method for extracting a meaningful subset of keywords from the triples represented in the knowledge graph.

While our goal in this paper is to generate a set of question answer pairs for a given entity in a knowledge graph, we train the RNN model for generating natural language questions from a sequence of keywords using an open domain Community Question Answering (CQA) data. This ensures that the same trained RNN can be used with different knowledge graphs.

The main contributions of our work can be summarized as follows:

- We propose a method for extracting triples and keywords from a knowledge graph for constructing question keywords and answer pairs.
- We formulate the problem of generating natural language questions from keywords as a sequence to sequence learning problem that performs 110.47 % (relative) better than existing template based approaches.
- We train our model using 1M questions from WikiAnswers thereby ensuring that it is not tied to any specific knowledge graph.
- Finally, we show that appending the automatically generated QA pairs to existing training data for training a state of the art QA system (Jonathan Berant, 2014) improves the performance of the QA system by 5.5 percent (relative).

The remainder of this paper is organized as follows. In next section, we describe related work, followed by a description of our overall approach for extracting keywords from triples and generating natural language question answer pairs from them. We then describe the experiments performed to evaluate our system and then end with concluding remarks.

2 Related Work

There is only very recent work around generation of question answer pairs from knowledge graph (Seyler et al., 2015). On the other hand, there are several works around question generation that have been proposed in past with different motivations. We first present a brief overview of the question generation techniques proposed in the literature along with their limitations and then discuss the work around generation of questions answer pairs from knowledge graph.

A number of papers have looked at the problem of generating vocabulary questions using WordNet (Miller et al., 1990) and distributional similarity techniques (Brown et al., 2005; Heilman and Eskenazi, 2007). There are numerous works in automatic question generation from text. Many proposed methods are *syntax based* methods that use the parse structure of sentences, identify key phrases and apply some known transformation rules to create questions (Ali et al., 2010; Kalady et al., 2010; Varga, 2010). Mannem et al. (2010) further use semantic role labeling for transformation rules. There are also *template based* methods proposed where a question template is a pre-defined text with placeholder variables to be replaced with content from source text. Cai et al. (2006) propose an XML markup language that is used to manually create question templates. This is sensitive to the performance of syntactic and semantic parsing. Heilman and Smith (2010) use a rule based approach to transform a declarative sentence into several candidate questions and then rank them using a logistic regression model. These approaches involve creating templates manually and thus require huge manual work and have low recall.

A problem that has been studied recently and is similar to our problem of generating questions using knowledge graph is that of generating questions from Web queries. The motivation here is to automatically generate questions from queries for community-based question answering services such as Yahoo! Answers and WikiAnswers. The idea was first suggested by (Lin, 2008) and further developed by (Zhao et al., 2011) and (Zheng et al., 2011). Both of these approaches are template based approaches where the templates are learnt using a huge question corpus along with query logs. Dror et al. (2013) further proposed a learning to rank based method to obtain grammatically

correct and diverse questions from a given query where the candidate questions are generated using the approach proposed by (Zhao et al., 2011). These approaches use millions of query question pairs to learn question templates and thus have better generalization performance compared to earlier methods where templates were learnt manually.

Recently, Seyler et al. (2015) proposed a method to generate natural language questions from knowledge graphs given a topic of interest. They also provide a method to estimate difficulty of generated questions. The generation of question is done by manually created template patterns and therefore is limited in application. In contrast we propose an RNN based method to learn generation of natural language questions from a set of keywords. The model can be trained using a dataset containing open domain keywords and question pairs.

3 Approach

In this section we propose an approach to generate Question Answer (QA) pairs for a given entity E . Let KG be the knowledge graph which contains information about various entities in the form of triples. A triple consists of a subject, a predicate and an object. Subjects and objects are nodes in the KG , which could represent a person, a place, an abstract concept or any physical entity. Predicates are edges in the KG . They define type of relationship between the subject and the object.

The framework to generate QA pairs consists two major modules. The first module, **Question Keywords and Answer Extractor**, is language independent and extracts required knowledge about the entity E from the KG . The second module is a language dependent **RNN based Natural Language Question Generator**. When fed with the information extracted from the first part it generates natural language QA pairs.

3.1 Question Keywords and Answer Extractor

Question keywords are keywords necessary to generate a question in natural language, or it could also be viewed as a concise representation of a natural language question. For example, to generate a QA pair for the entity *London*. We can generate a natural language question like *What is the capital city of United Kingdom?* with the keywords $\{Capital, City, United Kingdom\}$. Also

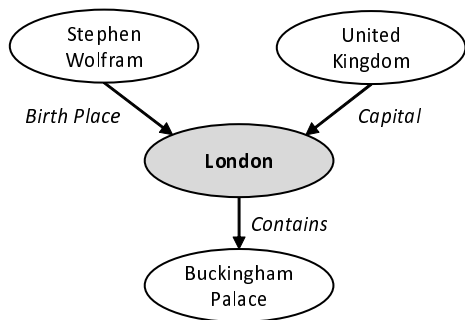


Figure 1: Triples with the entity *London* in a knowledge graph

since *London* is the answer to the above question, $(\{Capital, City, United Kingdom\}, London)$ together will form a Question Keyword and Answer (QKA) pair. One important note is that *Capital, City, United Kingdom* and *London* are the English labels of the node that represent these entities in the *KG*.

	Column A	Column B	Column C
Subject	United Kingdom	Stephen Wolfram	London
Domain	Country	Person	Location
Predicate	Capital	Birth Place	Contains
Object	London	London	Buckingham Palace
Range	City	Location	Location

Table 2: Examples of 5-tuples (subject, domain, predicate, object, range) for the entity *London*

In order to retrieve information about the given entity *E*, we need to first identify the node *n* that represents the entity *E* in the *KG*. One way to identify node *n* is to leverage the label (e.g. *rdfs:label*) property.

The next step is to retrieve all the neighbours of *n*. Let m_i be a neighbour of *n* in *KG*, connected by a predicate p_i . Here *i* is the index over all predicates whose subject or object is *n*. Figure 1 shows the entity *London* with three neighbours *United Kingdom*, *Stephen Wolfram* and *Buckingham Palace*. Each of these neighbours are related to *London* by a predicate. For example, *Stephen Wolfram* is related to *London* as it is his *Birth Place*.

Given a predicate p_i , let $sub(p_i)$ be the subject of p_i and $obj(p_i)$ be the object of p_i . A predicate is usually defined with a domain (subject type) and a range (object type) to provide better semantics. The domain and range defines the entity types that can be used as the subject and object of the predicate respectively. Let $domain(p_i)$ and $range(p_i)$ be the domain and range of p_i respectively. Let

$\{sub(p_i), domain(p_i), p_i, obj(p_i), range(p_i)\}$ be the 5-tuple associated with every p_i . Some examples of 5-tuples are shown column wise in Table 2.

We now describe how QKA pairs are extracted from 5-tuple. Let Q_k be the question keywords set and A_k be the answer to the question to be generated using Q_k . (Q_k, A_k) together will form a QKA pair. In this work, we consider only a single 5-tuple to generate a QKA pair. For example, we can generate QKA pair like $(\{Capital, City, United Kingdom\}, London)$ using *Column A* of Table 2. But we will not generate QKA pair like $(\{Capital, City, United Kingdom, Birth Place, Stephen Wolfram\}, London)$ using both *Column A & B* of Table 2.

We use the following rules to generate QKA pairs from 5-tuples.

- Unique Forward Relation :** If p_i is unique for $sub(p_i)$ in *KG*, then Q_k will include $sub(p_i)$, p_i and $range(p_i)$. A_k will be $obj(p_i)$. If p_i is not unique for $sub(p_i)$, then there could be multiple possible answers to the generated question including $obj(p_i)$, and therefore we do not generate such a QKA pair. When this is applied to *Column A* of Table 2, we generate $(\{Capital, City, United Kingdom\}, London)$ as a QKA pair. There is no QKA pair generated for *Column C* using this rule as *London* contains many locations like *Buckingham Palace*, *City of Westminster*, etc.
- Unique Reverse Relation :** If p_i is unique for $obj(p_i)$ in *KG*, then Q_k will include $obj(p_i)$, p_i and $domain(p_i)$. A_k is $sub(p_i)$. Similar to unique forward relation, this rule can be applied to *Column A* of Table 2 and cannot be applied to *Column B & C*.

3.2 RNN based Natural Language Question Generator

In the previous sub-section, we proposed an approach for creating *question keywords* and *answer* pairs. Now we propose a model for generating natural language questions from a given set of question keywords. We treat the keywords, $QK = \{qk_1, \dots, qk_m\}$, as an input sequence and the question, $Q = \{q_1, \dots, q_l\}$, as the output sequence. This design choice of treating a set of keywords as a sequence and not as a bag of words

allows us to generate different semantically valid questions from the same set K based on the order of the words in the set. For example, given the question keywords, $QK = \{King, Sweden\}$ we can generate two semantically valid questions by changing the order of King and Sweden: (i) Who is the King of Sweden? and (ii) Does Sweden have a King?

We propose a Natural Language Question Generation (NLQG) model that first encodes the input sequence using some distributed representation and then decodes the output sequence from this encoded representation. Specifically, we use a RNN based encoder and decoder recently proposed for language processing tasks by number of groups (Cho et al., 2014; Sutskever et al., 2014). We now formally define the encoder and decoder models.

Let m be the number of keywords in the input sequence. We represent each keyword using a fixed size vector $x_i \in \mathbb{R}^n$. The function of the encoder is to map this sequence of x_i 's to a fixed size encoding. We use a RNN to compute h_m using the following recursive equation:

$$h_i = \Phi(h_{i-1}, x_i), \quad (1)$$

where, $h_i \in \mathbb{R}^n$ is the hidden representation at position i . h_m is the final encoded hidden state vector for this sequence. We use LSTM units (Hochreiter and Schmidhuber, 1997) as Φ for our implementation based on its recent success in language processing tasks (Bahdanau et al., 2015).

The function of the decoder is to compute the probability of the output sequence $Q = \{q_1, \dots, q_l\}$ given the encoded vector h_m . Note that l is the length of the output sequence and may be different from m . This joint conditional probability of Q is decomposed into l conditional probabilities:

$$p(q_1, \dots, q_l | h_m) = \prod_{j=1}^l p(q_j | \{q_1, \dots, q_{j-1}\}, h_m). \quad (2)$$

Now we model $p(q_j | q_{<j}, h_m)$ at each position j by using a RNN decoder as follows:

$$p(q_j | q_{<j}, h_m) = \Theta(q_{j-1}, g_j, h_m), \quad (3)$$

where Θ is a non-linear function, that outputs the probability of q_j , and g_j is the hidden state of the decoder RNN.

To train this RNN model, we use a keyword sequence and question pairs generated from an open

domain Community Question Answering website. We provide more details on how the data is created and used for training in the experiments section.

At runtime, every permutation of the question keywords QK extracted is fed as input to the trained RNN. We pick the question Q with the highest probability of generation across all permutations, as the question generated from the question keywords QK .

4 Experiments

In this section we perform experiments to demonstrate how the proposed approach outperforms the existing template based approach for generating questions from the keywords. We also evaluate the quality of the QA pairs generated from knowledge graph.

4.1 Datasets

For training the K2Q-RNN model we require a set of keywords and question pairs. We use a large collection of open-domain questions available from WikiAnswers dataset². This dataset has around 20M questions. We randomly selected 1M questions from this corpus for training and 5k questions for testing (the maximum length of a question was restricted to 50 words). We extract keywords from the selected questions by retaining only Nouns, Verbs and Adjectives in the question. The parts of speech tags were identified using Stanford Tagger (Toutanova et al., 2003). We form an ordered sequence of keywords by retaining these extracted words in the same order in which they appear in the original question. This sequence of keywords along with the original question forms one input-output sequence pair for training.

4.2 Methods

We evaluate and compare the following methods:

K2Q-RNN: This is our approach proposed in the paper. For the encoder we use a bi-directional RNN containing one hidden layer of 1000 units. Each word in the input vocabulary is represented using a word vector which is randomly initialized and learnt during the training process. The decoder also contains one hidden layer comprising of 1000 units. At the output layer of the decoder a softmax function gives the distribution over the entire target vocabulary. We use the top 30,000

² Available at <http://knowitall.cs.washington.edu/oqa/data/wikianswers/>

most frequent words in the 1M training questions as the target vocabulary. If any sequence contains a word not belonging to this list then that word is mapped to a special token ([UNK]) that is also considered a part of the output vocabulary. We use a mini batch stochastic gradient descent algorithm together with Adadelta (Zeiler, 2012) to train our model. We used a mini-batch size of 50 and trained the model for 10 epochs. We used the beam search with the beam size to 12 to generate the question that approximately maximizes conditional probability defined in Equation 2.

K2Q-PBSMT: As mentioned earlier, we treat the problem of generating questions from keywords as a sequence to sequence translation problem. A Phrase Based Machine Translation System (PBSMT) can also be employed for this task by considering that the keyword sequences belong to a source language and the question sequences belong to a target language. We compare our approach with a standard phrase-based MT system, MOSES (Koehn et al., 2007) trained using the same 1M sequence pairs constructed from the WikiAnswers dataset. We used a 5-gram language model trained on the 1M target question sequences and tuned the parameters of the decoder using 1000 held-out sequence (these were held out from the 1M training pairs).

K2Q-Template: For template based approach we use the method proposed by (Zhao et al., 2011) along with the Word2Vec (Mikolov et al., 2015) ranking as proposed by (Raghu et al., 2015). The Word2Vec ranking provides better generalization than the ranking proposed by (Zhao et al., 2011). We learn the templates using the same 1M training pairs extracted from WikiAnswers.

4.3 Evaluation metrics

We evaluate the performance of K2Q RNN with other baselines to compare the K2Q approaches, we use BLEU score (Papineni et al., 2002) between the generated question and the reference question. BLEU score is typically used in evaluating the performance of MT systems and captures the average n-gram overlap between the generated sequence and the reference sequence. We consider n-grams upto length 4. BLEU score does not capture the true performance of the system. For example, if the trained model simply reproduces all keywords in the generated question then also the unigram overlap will be high resulting in a higher

Method	BLEU Score	Human Judgment accuracy (%)
K2Q-Template	25.58	28.57
K2Q-PBSMT	50.90	44.29
K2Q-RNN	50.14	60.13

Table 3: Automatic Evaluation (column 2): The BLEU scores of generated questions for the test set. Human Evaluation (Column 3): Percentage of perfect questions generated by *K2Q-Template*, *K2Q-PBSMT*, and *K2Q-RNN*

BLEU score. Further, we had only one reference question (ground truth) per test instance which is not sufficient to capture the different ways of expressing the question. In this case, BLEU score will be unnecessarily harsh on the model even if it generates a valid paraphrase of the reference question. To account for this we also perform a manual evaluation. We show the generated output to four human annotator and ask him/her to assign following ratings to the generated question, Rating 4 : Perfect without error, Rating 3 : Good with one error, missing/addition of article or preposition, but still meaningful, Rating 2 : Many errors, Rating 1 : Failure.

4.4 Results

4.4.1 RNN based Natural Language Question Generator

We first evaluate the performance of K2Q approaches using 5000 test instances from the WikiAnswers dataset. We extract the keyword sequence from these test questions using the same method described above. We compute the BLEU score by comparing the generated question with the original question. The results are presented in Table 3. Both K2Q-RNN and K2Q-PBSMT clearly outperform the template based method which shows that there is merit in formulating this problem as a sequence to sequence learning problem. To be sure that the results are not misleading due to some of the drawbacks of BLEU score as described earlier, we also do a manual evaluation. For this, we randomly selected 700 questions from the test set. We showed the questions generated by the three methods to different human annotators and asked them to assign a score of 1 to 4 to each question (based on the guidelines described earlier). The evaluators had no knowledge about the method used to generate each question shown to them. We only consider questions with rating 4 (perfect without any errors) for each

Ground truth	K2Q-PBSMT	K2Q-RNN
pitching in baseball ?	pitching in baseball ?	what is pitching in baseball ?
difference between mergercaqs and amalgamation ?	what is the difference between mergercaqs amalgamation ?	what is the difference between mergercaqs and amalgamation ?
did great britain control iraq ?	great britain control in iraq ?	how did the great britain control the iraq ?
what is the critical analysis of the poem a river ?	critical analysis of the poem a river ?	what is the most critical analysis of the poem a river ?
global warming affect population growth ?	global warming affect the population growth ?	can global warming affect the population growth ?

Table 4: Example questions from *Ground truth*, *K2Q-PBSMT* and *K2Q-RNN*

Entity	Keyword Query	Generated Question	Answer
Alan Turing	birth place alan turing	where is the birth place of alan turing ? (✓)	maida vale (✓)
	inventor lu decomposition	who was the inventor of lu decomposition ? (✓)	alan turing (✓)
	tv episodes alan turing	tv episodes of alan turing ? (✓)	dangerous knowledge (✓)
	author mathematical logic	what is the author of the mathematical logic ? (x)	alan turing (✓)
France	ioc code france	what is the ioc code for france ? (✓)	fr (✓)
	capital france	what is the capital of france ? (✓)	paris (✓)
	location lake annecy	what is the location of lake annecy ? (✓)	france (✓)
	albin haller country	is albin haller a country ? (x)	france (x)
Wimbledon	wimbledon first date occurrence	what was the wimbledon first date of occurrence ? (✓)	1877-07-09 (✓)
	current frequency wimbledon	what is the current frequency of wimbledon ? (✓)	yearly (✓)
	official website wimbledon	what is the official website for wimbledon ? (✓)	http://www.wimbledon.com/ (✓)

Table 5: Example question-answer pairs extracted for different entities by using *Freebase* and *K2Q-RNN*. Question-Answer pairs are considered correct if and only if both are marked with ✓ by human judges.

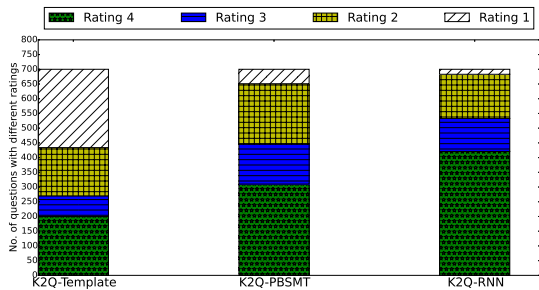


Figure 2: Ratings given by human judges for generated questions for *K2Q-Template*, *K2Q-PBSMT* and *K2Q-RNN* (Best viewed in color).

method and calculate the accuracy, shown in Table 3. Figure 2 shows the distribution of ratings assigned by the annotators. Once gain *K2Q-RNN* and *K2Q-PBSMT* outperform the template based approach. Further, the human evaluation shows *K2Q-RNN* performs better than *K2Q-PBSMT*. Table 4 shows example questions that may have a high BLEU score for *K2Q-PBSMT*, however the *K2Q-RNN* has a better human judgement.

Next, we also compare the performance of these methods for input keyword sequences of different lengths. For this, we consider all test instances having k keywords and mark the generated question as correct if it was given a rating of 4 by the human annotator. The results of this experiment are plotted in Figure 3 where the x-axis represents number of keywords and y-axis represents the percentage of test instances for which correct (rating 4) questions were generated. Once again we see that *K2Q-RNN* clearly outperforms the other methods at all input sequence sizes.

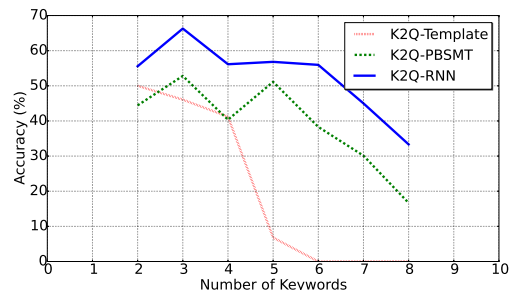


Figure 3: The plot shows the performance of 3 methods *K2Q-Template*, *K2Q-PBSMT*, and *K2Q-RNN* as a functions of number of keywords.

4.4.2 Generating Question-Answers pairs from Freebase

In this section we describe the performance of *K2Q-RNN* for generating QA pairs from a Knowledge Graph. For our evaluation purpose, we use *Freebase* as the Knowledge Graph. We randomly picked 27 *Freebase* entities of various types (person, location, organization, etc) and extracted all 5-tuples containing them. To create a diverse QA pairs we retained only two instances (5-tuples) for each predicate or relation type. Some predicates (like summary, quotations) have long text as their objects, some predicates (like Daylife Topic ID, Hero image ID) are difficult for annotator to validate. So, we filtered the list further by removing above mentioned predicates and generated a total of 485 QKA pairs. We manually evaluated these generated QA pairs and marked them as correct only if generated question along with the answer together convey the information represented

Train Sources	Train	Test	F1-score(%)
WQ	3778	2032	39.9
WQ+GQA	11334	2032	42.1
WQ+GT	11334	2032	43.6

Table 6: PARASEMPRE Evaluation: WQ=WebQuestions, GQA=Generated QA pairs from *SimpleQuestions* test dataset, GT=Ground truth data from *SimpleQuestions* test dataset.

in the 5-tuple. A few QA pairs were marked correct by the annotators, even though the question was not grammatically correct but convey the right intent. Some examples of such questions are *melting point of propyl alcohol?*, *stanford university student radio station?*. Overall, **33.61%** of the QA pairs generated by our method were annotated correct. Table 5 shows some correct and incorrect QA pairs generated by our method.

4.4.3 Extrinsic Evaluation

As an extrinsic evaluation of the quality of our QA generation model, we use QA pairs generated by our model to improve the performance of a state of the art QA system called PARASEMPRE (Jonathan Berant, 2014). PARASEMPRE is a semantic parser, which maps natural language questions to intermediate logical forms which in turn are used to answer the question. The standard training set used for training PARASEMPRE is a part of the *WebQuestions* and contains 3778 QA pairs. We appended this train with 7556 automatically generated QA pairs (resulting in tripling of the training set). Table 6 then compares the same system trained on the following different training sets: (i) Only Web Questions (WQ) dataset (ii) WQ + Generated Question Answers (GQA) and (iii) WQ + Ground Truth (GT) QA pairs. The GT QA pairs were obtained from the *SimpleQuestions* (Bordes et al., 2015) test data and have a one-to-one correspondence to the GQA data (hence the results are comparable). We see a relative improvement of 5.5% in the *F1-score* of the system by adding GQA. Further, the performance gains are comparable to those obtained by using GT QA pairs.

4.4.4 Error analysis

We inspected all the QA pairs generated by our method to identify some common mistakes. We found that most errors corresponded to (i) con-

fusion between is/are and do/does (ii) incorrect use of determiners (missing articles, confusion between a/the and addition of extra articles). Another problem occurs when the extracted keyword sequence contains a stop word. This happens when dealing with triples such as (*{also known as, Andre Agassi}*, *Agassi*). Since, during training we retain only content words (nouns, adjectives, verbs) in the input sequence, the model fails to deal with such stop words at test time and simply produces unknown token (UNK) in the output. Another set of errors corresponds to mismatch between the subject type and question type. For example, we observed that in a few cases, the model incorrectly generates a *what question* instead of a *who question* when the answer type is a person.

5 Conclusions

In this paper we propose a method for generating QA pairs for an given entity using a knowledge graph. We also propose an RNN based approach for generating natural language questions from an input keyword sequence. The proposed method performs significantly better than previously proposed template based method. We also do an extrinsic evaluation to show that the generated QA pairs help in improving the performance of a downstream QA system. In future, we plan to extend this work to support predicates with stop words and support predicates in various tenses.

References

- Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.
- Jonathan C Brown, Gwen A Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 819–826. Association for Computational Linguistics.
- Zhiqiang Cai, Vasile Rus, Hyun-Jeong Joyce Kim, Suresh C Susarla, Pavan Karnam, and Arthur C

- Graesser. 2006. Nlxml: A markup language for question generation. In *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, volume 2006, pages 2747–2752.
- KyungHyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.
- Gideon Dror, Yoelle Maarek, Avihai Mejer, and Idan Szpektor. 2013. From query to question in one click: suggesting synthetic questions to searchers. In *Proceedings of the 22nd international conference on World Wide Web*, pages 391–402. International World Wide Web Conferences Steering Committee.
- David A Ferrucci. 2012. Introduction to this is watson. *IBM Journal of Research and Development*, 56(3.4):1–1.
- Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu, and Jun Zhao. 2014. Question answering over linked data using first-order logic. In *Proceedings of Empirical Methods in Natural Language Processing*.
- Michael Heilman and Maxine Eskenazi. 2007. Application of automatic thesaurus extraction for computer generation of vocabulary questions. In *SLaTE*, pages 65–68.
- Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Percy Liang Jonathan Berant. 2014. Semantic parsing via paraphrasing. In *ACL*.
- Saidalavi Kalady, Ajeesh Elikkotttil, and Rajarshi Das. 2010. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 1–10. questiongeneration.org.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, ACL '07*, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Chin-Yew Lin. 2008. Automatic question generation from queries. In *Workshop on the Question Generation Shared Task*, pages 156–164.
- Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2015. Efficient estimation of word representations in vector space. In *ICLR*.
- George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography*, 3(4):235–244.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.
- Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive qa system. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 335.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. pages 696–699.
- Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 113–114. International World Wide Web Conferences Steering Committee.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.
- Andrea Varga. 2010. Le an ha 2010 wlv: A question generation system for the qgstec 2010 task b. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 80–83.
- P.J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct.

- Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1107–1116. ACM.
- Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arxiv*, 1212.5701.
- Shiqi Zhao, Haifeng Wang, Chao Li, Ting Liu, and Yi Guan. 2011. Automatically generating questions from queries for community-based question answering. In *IJCNLP*, pages 929–937.
- Zhicheng Zheng, Xiance Si, Edward Y Chang, and Xiaoyan Zhu. 2011. K2q: Generating natural language questions from keywords with user refinements. In *IJCNLP*, pages 947–955.
- Lei Zou, Ruizhe Huang, Haixun Wang, Jeffer Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM.

Enumeration of Extractive Oracle Summaries

Tsutomu Hirao and Masaaki Nishino and Jun Suzuki and Masaaki Nagata

NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237, Japan
{hirao.tsutomu,nishino.masaaki}@lab.ntt.co.jp
{suzuki.jun,nagata.masaaki}@lab.ntt.co.jp

Abstract

To analyze the limitations and the future directions of the extractive summarization paradigm, this paper proposes an Integer Linear Programming (ILP) formulation to obtain *extractive oracle summaries* in terms of ROUGE_n. We also propose an algorithm that enumerates all of the oracle summaries for a set of reference summaries to exploit F-measures that evaluate which system summaries contain how many sentences that are extracted as an oracle summary. Our experimental results obtained from Document Understanding Conference (DUC) corpora demonstrated the following: (1) room still exists to improve the performance of extractive summarization; (2) the F-measures derived from the enumerated oracle summaries have significantly stronger correlations with human judgment than those derived from single oracle summaries.

1 Introduction

Recently, compressive and abstractive summarization are attracting attention (e.g., Almeida and Martins (2013), Qian and Liu (2013), Yao et al. (2015), Banerjee et al. (2015), Bing et al. (2015)). However, extractive summarization remains a primary research topic because the linguistic quality of the resultant summaries is guaranteed, at least at the sentence level, which is a key requirement for practical use (e.g., Hong and Nenkova (2014), Hong et al. (2015), Yogatama et al. (2015), Parveen et al. (2015)).

The summarization research community is experiencing a paradigm shift from extractive to compressive or abstractive summarization. Currently our question is: “Is extractive summariza-

tion still useful research?” To answer it, the ultimate limitations of the extractive summarization paradigm must be comprehended; that is, we have to determine its upper bound and compare it with the performance of the state-of-the-art summarization methods. Since ROUGE_n is the de-facto automatic evaluation method and is employed in many text summarization studies, an oracle summary is defined as a set of sentences that have a maximum ROUGE_n score. If the ROUGE_n score of an oracle summary outperforms that of a system that employs another summarization approach, the extractive summarization paradigm is worthwhile to leverage research resources.

As another benefit, identifying an oracle summary for a set of reference summaries allows us to utilize yet another evaluation measure. Since both oracle and extractive summaries are sets of sentences, it is easy to check whether a system summary contains sentences in the oracle summary. As a result, F-measures, which are available to evaluate a system summary, are useful for evaluating classification-based extractive summarization (Mani and Bloedorn, 1998; Osborne, 2002; Hirao et al., 2002). Since ROUGE_n evaluation does not identify which sentence is important, an F-measure conveys useful information in terms of “important sentence extraction.” Thus, combining ROUGE_n and an F-measure allows us to scrutinize the failure analysis of systems.

Note that more than one oracle summary might exist for a set of reference summaries because ROUGE_n scores are based on the unweighted counting of n-grams. As a result, an F-measure might not be identical among multiple oracle summaries. Thus, we need to enumerate the oracle summaries for a set of reference summaries and compute the F-measures based on them.

In this paper, we first derive an Integer Linear Programming (ILP) problem to extract an oracle

summary from a set of reference summaries and a source document(s). To the best of our knowledge, this is the first ILP formulation that extracts oracle summaries. Second, since it is difficult to enumerate oracle summaries for a set of reference summaries using ILP solvers, we propose an algorithm that efficiently enumerates all oracle summaries by exploiting the branch and bound technique. Our experimental results on the Document Understanding Conference (DUC) corpora showed the following:

1. Room still exists for the further improvement of extractive summarization, *i.e.*, where the ROUGE_n scores of the oracle summaries are significantly higher than those of the state-of-the-art summarization systems.
2. The F-measures derived from multiple oracle summaries obtain significantly stronger correlations with human judgment than those derived from single oracle summaries.

2 Definition of Extractive Oracle Summaries

We first briefly describe ROUGE_n. Given set of reference summaries \mathbf{R} and system summary S , ROUGE_n is defined as follows:

$$\text{ROUGE}_n(\mathbf{R}, S) = \frac{|\mathbf{R}| \sum_{j=1}^{|U(\mathcal{R}_k)|} \min\{N(g_j^n, \mathcal{R}_k), N(g_j^n, S)\}}{\sum_{k=1}^{|\mathbf{R}|} \sum_{j=1}^{|U(\mathcal{R}_k)|} N(g_j^n, \mathcal{R}_k)}. \quad (1)$$

\mathcal{R}_k denotes the multiple set of n-grams that occur in k -th reference summary R_k , and \mathcal{S} denotes the multiple set of n-grams that appear in system-generated summary S (a set of sentences). $N(g_j^n, \mathcal{R}_k)$ and $N(g_j^n, \mathcal{S})$ return the number of occurrences of n-gram g_j^n in the k -th reference and system summaries, respectively. Function $U(\cdot)$ transforms a multiple set into a normal set. ROUGE_n takes values in the range of $[0, 1]$, and when the n-gram occurrences of the system summary agree with those of the reference summary, the value is 1.

In this paper, we focus on extractive summarization, employ ROUGE_n as an evaluation measure,

and define the oracle summaries as follows:

$$O = \arg \max_{S \subseteq D} \text{ROUGE}_n(\mathbf{R}, S) \quad (2)$$

s.t. $\ell(S) \leq L_{\max}$.

D is the set of all the sentences contained in the input document(s), and L_{\max} is the length limitation of the oracle summary. $\ell(S)$ indicates the number of words in the system summary. Eq. (2) is an NP-hard combinatorial optimization problem, and no polynomial time algorithms exist that can attain an optimal solution.

3 Related Work

Lin and Hovy (2003) utilized a naive exhaustive search method to obtain oracle summaries in terms of ROUGE_n and exploited them to understand the limitations of extractive summarization systems. Ceylan et al. (2010) proposed another naive exhaustive search method to derive a probability density function from the ROUGE_n scores of oracle summaries for the domains to which source documents belong. The computational complexity of naive exhaustive methods is exponential to the size of the sentence set. Thus, it may be possible to apply them to single document summarization tasks involving a dozen sentences, but it is infeasible to apply them to multiple document summarization tasks that involve several hundred sentences.

To describe the difference between the ROUGE_n scores of oracle and system summaries in multiple document summarization tasks, Riedhammer et al. (2008) proposed an approximate algorithm with a genetic algorithm (GA) to find oracle summaries. Moen et al. (2014) utilized a greedy algorithm for the same purpose. Although GA or greedy algorithms are widely used to solve NP-hard combinatorial optimization problems, the solutions are not always optimal. Thus, the summary does not always have a maximum ROUGE_n score for the set of reference summaries. Both works called the summary found by their methods the oracle, but it differs from the definition in our paper.

Since summarization systems cannot reproduce human-made reference summaries in most cases, oracle summaries, which can be reproduced by summarization systems, have been used as training data to tune the parameters of summarization systems. For example, Kulesza and Tasker (2011) and Sipos et al. (2012) trained their summarizers

with oracle summaries found by a greedy algorithm. Peyrard and Eckle-Kohler (2016) proposed a method to find a summary that approximates a ROUGE score based on the ROUGE scores of individual sentences and exploited the framework to train their summarizer. As mentioned above, such summaries do not always agree with the oracle summaries defined in our paper. Thus, the quality of the training data is suspect. Moreover, since these studies fail to consider that a set of reference summaries has multiple oracle summaries, the score of the loss function defined between their oracle and system summaries is not appropriate in most cases.

As mentioned above, no known efficient algorithm can extract “exact” oracle summaries, as defined in Eq. (2), *i.e.*, because only a naive exhaustive search is available. Thus, such approximate algorithms as a greedy algorithm are mainly employed to obtain them.

4 Oracle Summary Extraction as an Integer Linear Programming (ILP) Problem

To extract an oracle summary from document(s) and a given set of reference summaries, we start by deriving an Integer Linear Programming (ILP) problem. Since the denominator of Eq. (1) is constant for a given set of reference summaries, we can find an oracle summary by maximizing the numerator of Eq. (1). Thus, the ILP formulation is defined as follows:

$$\underset{z}{\text{maximize}} \quad \sum_{k=1}^{|\mathcal{R}|} \sum_{j=1}^{|U(\mathcal{R}_k)|} z_{kj} \quad (3)$$

$$s.t. \quad \sum_{i=1}^{|D|} \ell(s_i) x_i \leq L_{\max} \quad (4)$$

$$\forall j : \sum_{i=1}^{|D|} N(g_j^n, s_i) x_i \geq z_{kj} \quad (5)$$

$$\forall j : N(g_j^n, \mathcal{R}_k) \geq z_{kj} \quad (6)$$

$$\forall i : x_i \in \{0, 1\} \quad (7)$$

$$\forall j : z_{kj} \in \mathbb{Z}_+. \quad (8)$$

Here, z_{kj} is the count of the j -th n -gram of the k -th reference summary in the oracle summary, *i.e.*, $z_{kj} = \min\{N(g_j^n, \mathcal{R}_k), N(g_j^n, \mathcal{S})\}$. $\ell(\cdot)$ returns the number of words in the sentence, x_i is a binary indicator, and $x_i = 1$ denotes that the i -th sentence s_i is included in

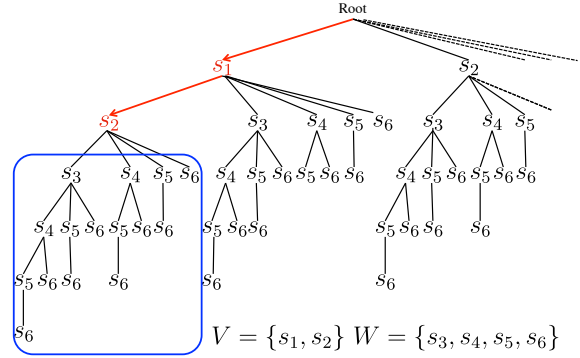


Figure 1: Example of a search tree

the oracle summary. $N(g_j^n, s_i)$ returns the number of occurrences of n -gram g_j^n in the i -th sentence. Constraints (5) and (6) ensure that $z_{kj} = \min\{N(g_j^n, \mathcal{R}_k), N(g_j^n, \mathcal{S})\}$.

5 Branch and Bound Technique for Enumerating Oracle Summaries

Since enumerating oracle summaries with an ILP solver is difficult, we extend the exhaustive search approach by introducing a search and prune technique to enumerate the oracle summaries. The search pruning decision is made by comparing the current upper bound of the ROUGE_n score with the maximum ROUGE_n score in the search history.

5.1 ROUGE_n Score for Two Distinct Sets of Sentences

The enumeration of oracle summaries can be regarded as a depth-first search on a tree whose nodes represent sentences. Fig. 1 shows an example of a search tree created in a naive exhaustive search. The nodes represent sentences and the path from the root node to an arbitrary node represents a summary. For example, the red path in Fig. 1 from the root node to node s_2 represents a summary consisting of sentences s_1, s_2 . By utilizing the tree, we can enumerate oracle summaries by exploiting depth-first searches while excluding the summaries that violate length constraints. However, this naive exhaustive search approach is impractical for large data sets because the number of nodes inside the tree is $2^{|D|}$.

If we prune the unwarranted subtrees in each step of the depth-first search, we can make the search more efficient. The decision to search or prune is made by comparing the current upper

bound of the ROUGE_n score with the maximum ROUGE_n score in the search history. For instance, in Fig. 1, we reach node s_2 by following this path: “Root $\rightarrow s_1, \rightarrow s_2$ ”. If we estimate the maximum ROUGE_n score (upper bound) obtained by searching for the descendant of s_2 (the subtree in the blue rectangle), we can decide whether the depth-first search should be continued. When the upper bound of the ROUGE_n score exceeds the current maximum ROUGE_n in the search history, we have to continue. When the upper bound is smaller than the current maximum ROUGE_n score, no summary is optimal that contains s_1, s_2 , so we can skip subsequent search activity on the subtree and proceed to check the next branch: “Root $\rightarrow s_1 \rightarrow s_3$ ”.

To estimate the upper bound of the ROUGE_n score, we re-define it for two distinct sets of sentences, V and W , *i.e.*, $V \cap W = \phi$, as follows:

$$\begin{aligned} \text{ROUGE}_n(\mathbf{R}, V \cup W) &= \text{ROUGE}_n(\mathbf{R}, V) \\ &+ \text{ROUGE}'_n(\mathbf{R}, V, W). \end{aligned} \quad (9)$$

Here ROUGE'_n is defined as follows:

$$\begin{aligned} \text{ROUGE}'_n(\mathbf{R}, V, W) &= \\ \frac{\sum_{k=1}^{|\mathbf{R}|} \sum_{t_n \in U(\mathcal{R}_k)} \min\{N(t_n, \mathcal{R}_k \setminus \mathcal{V}), N(t_n, \mathcal{W})\}}{\sum_{k=1}^{|\mathbf{R}|} \sum_{t_n \in U(\mathcal{R}_k)} N(t_n, \mathcal{R}_k)}. \end{aligned} \quad (10)$$

\mathcal{V}, \mathcal{W} are the multiple sets of n -grams found in the sets of sentences V and W , respectively.

Theorem 1. *Eq. (9) is correct.*

We omit the proof of Theorem 1 due to space limitations.

5.2 Upper Bound of ROUGE_n

Let V be the set of sentences on the path from the current node to the root node in the search tree, and let W be the set of sentences that are the descendants of the current node. In Fig. 1, $V = \{s_1, s_2\}$ and $W = \{s_3, s_4, s_5, s_6\}$. According to Theorem 1, the upper bound of the ROUGE_n score is defined as:

$$\begin{aligned} \widehat{\text{ROUGE}}_n(\mathbf{R}, V) &= \text{ROUGE}_n(\mathbf{R}, V) + \\ \max_{\Omega \subseteq W} \{ \text{ROUGE}'_n(\mathbf{R}, V, \Omega) : \ell(\Omega) \leq L_{\max} - \ell(V) \} \end{aligned} \quad (11)$$

Algorithm 1 Algorithm to Find Upper Bound of ROUGE_n

```

1: Function:  $\widehat{\text{ROUGE}}_n(\mathbf{R}, V)$ 
2:    $W \leftarrow \text{descendant}(\text{last}(V)), W' \leftarrow \phi$ 
3:    $U \leftarrow \text{ROUGE}(\mathbf{R}, V)$ 
4:   for each  $w \in W$  do
5:      $\text{append}(W', \frac{\text{ROUGE}'_n(\mathbf{R}, V, \{w\})}{\ell(w)})$ 
6:   end for
7:    $\text{sort}(W', \text{'descend'})$ 
8:   for each  $w \in W'$  do
9:     if  $L_{\max} - \ell(\{w\}) \geq 0$  then
10:       $U \leftarrow U + \text{ROUGE}'_n(\mathbf{R}, V, \{w\})$ 
11:       $L_{\max} \leftarrow L_{\max} - \ell(\{w\})$ 
12:     else
13:       $U \leftarrow U + \frac{\text{ROUGE}'_n(\mathbf{R}, V, \{w\})}{\ell(\{w\})} \times L_{\max}$ 
14:      break the loop
15:     end if
16:   end for
17:   return  $U$ 
18: end

```

Since the second term on the right side in Eq. (11) is an NP-hard problem, we turn to the following relation by introducing inequality, $\text{ROUGE}'_n(\mathbf{R}, V, \Omega) \leq \sum_{\omega \in \Omega} \text{ROUGE}'_n(\mathbf{R}, V, \{\omega\})$,

$$\begin{aligned} \max_{\Omega \subseteq W} \{ \text{ROUGE}'_n(\mathbf{R}, V, \Omega) : \ell(\Omega) \leq L_{\max} - \ell(V) \} \\ \leq \max_{\mathbf{x}} \left\{ \sum_{i=1}^{|\mathcal{W}|} \text{ROUGE}'_n(\mathbf{R}, V, \{w_i\}) x_i : \right. \\ \left. \sum_{i=1}^{|\mathcal{W}|} \ell(\{w_i\}) x_i \leq L_{\max} - \ell(V) \right\}. \end{aligned} \quad (12)$$

Here, $\mathbf{x} = (x_1, \dots, x_{|\mathcal{W}|})$ and $x_i \in \{0, 1\}$. The right side of Eq. (12) is a knapsack problem, *i.e.*, a 0-1 ILP problem. Although we can obtain the optimal solution for it using dynamic programming or ILP solvers, we solve its linear programming relaxation version by applying a greedy algorithm for greater computation efficiency. The solution output by the greedy algorithm is optimal for the relaxed problem. Since the optimal solution of the relaxed problem is always larger than that of the original problem, the relaxed problem solution can be utilized as the upper bound. Algorithm 1 shows the pseudocode that attains the upper bound of ROUGE_n . In the algorithm, U indicates the upper bound score of ROUGE_n . We first set the initial score of upper bound U to $\text{ROUGE}_n(\mathbf{R}, V)$ (line 3). Then we compute the density of the ROUGE'_n scores ($\text{ROUGE}'_n(\mathbf{R}, V, \{w\})/\ell(w)$) for each sentence w in W and sort them in descending order (lines 4 to 6). When we have room to add w to the summary, we update U by adding the $\text{ROUGE}'_n(\mathbf{R}, V, \{w\})$ (line 10) and update length

Algorithm 2 Greedy algorithm to obtain initial score

```

1: Function: GREEDY( $\mathbf{R}, D, L_{\max}$ )
2:  $L \leftarrow 0, S \leftarrow \phi, E \leftarrow D$ 
3: while  $E \neq \phi$  do
4:    $s^* \leftarrow \arg \max_{s \in E} \left\{ \frac{\text{ROUGE}_n(\mathbf{R}, S \cup \{s\}) - \text{ROUGE}_n(\mathbf{R}, S)}{\ell(\{s\})} \right\}$ 
5:    $L \leftarrow L + \ell(\{s^*\})$ 
6:   if  $L \leq L_{\max}$  then
7:      $S \leftarrow S \cup \{s^*\}$ 
8:   end if
9:    $E \leftarrow E \setminus \{s^*\}$ 
10: end while
11:  $i^* \leftarrow \arg \max_{i \in D, \ell(\{i\}) \leq L_{\max}} \text{ROUGE}_n(\mathbf{R}, \{i\})$ 
12:  $S^* \leftarrow \arg \max_{K \subseteq \{i^*\}, S} \text{ROUGE}_n(\mathbf{R}, K)$ 
13: return  $\text{ROUGE}_n(\mathbf{R}, S^*)$ 
14: end

```

constraint L_{\max} (line 11). When we do not have room to add w , we update U by adding the score obtained by multiplying the density of w by the remaining length, L_{\max} (line 13), and exit the while loop.

5.3 Initial Score for Search

Since the branch and bound technique prunes the search by comparing the best solution found so far with the upper bounds, obtaining a good solution in the early stage is critical for raising search efficiency.

Since ROUGE_n is a monotone submodular function (Lin and Bilmes, 2011), we can obtain a good approximate solution by a greedy algorithm (Khuller et al., 1999). It is guaranteed that the score of the obtained approximate solution is larger than $\frac{1}{2}(1 - \frac{1}{e})\text{OPT}$, where OPT is the score of the optimal solution. We employ the solution as the initial ROUGE_n score of the candidate oracle summary.

Algorithm 2 shows the greedy algorithm. In it, S denotes a summary and D denotes a set of sentences. The algorithm iteratively adds sentence s^* that yields the largest gain in the ROUGE_n score to current summary S , provided the length of the summary does not violate length constraint L_{\max} (line 4). After the while loop, the algorithm compares the ROUGE_n score of S with the maximum ROUGE_n score of the single sentence and outputs the larger of the two scores (lines 11 to 13).

5.4 Enumeration of Oracle summaries

By introducing threshold τ as the best ROUGE_n score in the search history, pruning decisions involve the following three conditions:

Algorithm 3 Branch and bound technique to enumerate oracle summaries

```

1: Read  $\mathbf{R}, D, L_{\max}$ 
2:  $\tau \leftarrow \text{GREEDY}(\mathbf{R}, D, L_{\max}), O_\tau \leftarrow \phi$ 
3: for each  $s \in D$  do
4:   append( $S, (\text{ROUGE}_n(\mathbf{R}, \{s\}), s)$ )
5: end for
6: sort( $S, \text{'descend'}$ )
7: call FINDORACLE( $S, C$ )
8: output  $O_\tau$ 
9: Procedure: FINDORACLE( $Q, V$ )
10: while  $Q \neq \phi$  do
11:    $s \leftarrow \text{shift}(Q)$ 
12:   append( $V, s$ )
13:   if  $L_{\max} - \ell(V) \geq 0$  then
14:     if  $\text{ROUGE}_n(\mathbf{R}, V) \geq \tau$  then
15:        $\tau \leftarrow \text{ROUGE}_n(\mathbf{R}, V)$ 
16:       append( $O_\tau, V$ )
17:       call FINDORACLE( $Q, V$ )
18:     else if  $\widehat{\text{ROUGE}}_n(\mathbf{R}, V) \geq \tau$  then
19:       call FINDORACLE( $Q, V$ )
20:     end if
21:   end if
22:   pop( $V$ )
23: end while
24: end

```

1. $\text{ROUGE}_n(\mathbf{R}, V) \geq \tau$;
2. $\text{ROUGE}_n(\mathbf{R}, V) < \tau, \widehat{\text{ROUGE}}_n(\mathbf{R}, V) < \tau$;
3. $\text{ROUGE}_n(\mathbf{R}, V) < \tau, \widehat{\text{ROUGE}}_n(\mathbf{R}, V) \geq \tau$.

With case 1, we update the oracle summary as V and continue the search. With case 2, because both $\text{ROUGE}_n(\mathbf{R}, V)$ and $\widehat{\text{ROUGE}}_n(\mathbf{R}, V)$ are smaller than τ , the subtree whose root node is the current node (last visited node) is pruned from the search space, and we continue the depth-first search from the neighbor node. With case 3, we do not update oracle summary as V because $\text{ROUGE}_n(\mathbf{R}, V)$ is less than τ . However, we might obtain a better oracle summary by continuing the depth-first search because the upper bound of the ROUGE_n score exceeds τ . Thus, we continue to search for the descendants of the current node.

Algorithm 3 shows the pseudocode that enumerates the oracle summaries. The algorithm reads a set of reference summaries \mathbf{R} , length limitation L_{\max} , and set of sentences D (line 1) and initializes threshold τ as the ROUGE_n score obtained by the greedy algorithm (Algorithm 2). It also initializes O_τ , which stores oracle summaries whose ROUGE_n scores are τ , and priority queue C , which stores the history of the depth-first search (line 2). Next, the algorithm computes the ROUGE_n score for each sentence and stores S after sorting them in descending order. After that, we start a depth-first search by recursively call-

Year	Topics	Docs.	Sents.	Words	Refs.	Length
01	30	10	365	7706	89	100
02	59	10	238	4822	116	100
03	30	10	245	5711	120	100
04	50	10	218	4870	200	100
05	50	29.5	885	18273.5	300	250
06	50	25	732.5	15997.5	200	250
07	45	25	516	11427	180	250

Table 1: Statistics of data set

ing procedure FINDORACLE. In the procedure, we extract the top sentence from priority queue Q and append it to priority queue V (lines 11 to 12). When the length of V is less than L_{\max} , if $\text{ROUGE}_n(\mathbf{R}, V)$ is larger than threshold τ (case 1), we update τ as the score and append current V to O_τ . Then we continue the depth-first search by calling the procedure the FINDORACLE (lines 15 to 17). If $\widehat{\text{ROUGE}}_n(\mathbf{R}, V)$ is larger than τ (case 3), we do not update τ and O_τ but reenter the depth-first search by calling the procedure again (lines 18 to 19). If neither case 1 nor case 3 is true, we delete the last visited sentence from V and return to the top of the recurrence.

6 Experiments

6.1 Experimental Setting

We conducted experiments on the corpora developed for a multiple document summarization task in DUC 2001 to 2007. Table 1 show the statistics of the data. In particular, the DUC-2005 to -2007 data sets not only have very large numbers of sentences and words but also a long target length (the reference summary length) of 250 words.

All the words in the documents were stemmed by Porter’s stemmer (Porter, 1980). We computed ROUGE_1 scores, excluding stopwords, and computed ROUGE_2 scores, keeping them. Owczarzak et al. (2012) suggested using ROUGE_1 and keeping stopwords. However, as Takamura et al. argued (Takamura and Okumura, 2009), the summaries optimized with non-content words failed to consider the actual quality. Thus, we excluded stopwords for computing the ROUGE_1 scores.

We enumerated the following two types of oracle summaries: those for a set of references for a given topic and those for each reference in the set of references.

6.2 Results and Discussion

6.2.1 Impact of Oracle ROUGE_n scores

Table 2 shows the average $\text{ROUGE}_{1,2}$ scores of the oracle summaries obtained from both a set of references and each reference in the set (“multi” and “single”), those of the best conventional system (Peer), and those obtained from summaries produced by a greedy algorithm (Algorithm 2).

Oracle (single) obtained better $\text{ROUGE}_{1,2}$ scores than Oracle (multi). The results imply that it is easier to optimize a reference summary than a set of reference summaries. On the other hand, the $\text{ROUGE}_{1,2}$ scores of these oracle summaries are significantly higher than those of the best systems. The best systems obtained ROUGE_1 scores from 60% to 70% in “multi” and from 50% to 60% in “single” as well as ROUGE_2 scores from 40% to 55% in “multi” and from 30% to 40% in “single” for their oracle summaries.

Since the systems in Table 2 were developed over many years, we compared the ROUGE_n scores of the oracle summaries with those of the current state-of-the-art systems using the DUC-2004 corpus and obtained summaries generated by different systems from a public repository¹ (Hong et al., 2014). The repository includes summaries produced by the following seven state-of-the-art summarization systems: CLASSY04 (Conroy et al., 2004), CLASSY11 (Conroy et al., 2011), Submodular (Lin and Bilmes, 2012), DPP (Kulesza and Tasker, 2011), RegSum (Hong and Nenkova, 2014), OCCAMS_V (Davie et al., 2012; Conroy et al., 2013), and ICSISumm (Gillick and Favre, 2009; Gillick et al., 2009). Table 3 shows the results.

Based on the results, RegSum (Hong and Nenkova, 2014) achieved the best $\text{ROUGE}_1=0.331$ result, while ICSISumm (Gillick and Favre, 2009; Gillick et al., 2009) (a compressive summarizer) achieved the best result with $\text{ROUGE}_2=0.098$. These systems outperformed the best systems (Peers 65 and 67 in Table 2), but the differences in the ROUGE_n scores between the systems and the oracle summaries are still large. More recently, Hong et al. (2015) demonstrated that their system’s combination approach achieved the current best ROUGE_2 score, 0.105, for the DUC-2004 corpus. However, a large difference remains between the ROUGE_2 score of oracle and

¹<http://www.cis.upenn.edu/~nlp/corpora/sumrepo.html>

	01		02		03		04		05		06		07	
	R ₁	R ₂	R ₁	R ₂	R ₁	R ₂	R ₁	R ₂	R ₁	R ₂	R ₁	R ₂	R ₁	R ₂
Oracle (multi)	.400	.164	.452	.186	.434	.185	.427	.162	.445	.177	.491	.211	.506	.236
Oracle (single)	.500	.226	.515	.225	.525	.258	.519	.228	.574	.279	.607	.303	.622	.330
Greedy	.387	.161	.438	.184	.424	.182	.412	.157	.430	.173	.473	.206	.495	.234
Peer	.251	.080	.269	.080	.295	.094	.305	.092	.262	.073	.305	.095	.363	.117
ID	T	T	19	19	26	13	67	65	10	15	23	24	29	15

Table 2: ROUGE_{1,2} scores of oracle summaries, greedy summaries, and system summaries for each data set

System	ROUGE ₁	ROUGE ₂		
			single	multi
Oracle (multi)	.427	.162	ROUGE ₁	.451 .419
Oracle (single)	.519	.228	ROUGE ₂	.536 .530
CLASSY04	.305	.0897		
CLASSY11	.286	.0919		
Submodular	.300	.0933		
DPP	.309	.0960		
RegSum	.331	.0974		
OCCAMS_V	.300	.0974		
ICSISumm	.310	.0980		

Table 3: ROUGE_{1,2} scores for state-of-the-art summarization systems on DUC-2004 corpus

their summaries.

In short, the ROUGE_n scores of the oracle summaries are significantly higher than those of the current state-of-the-art summarization systems, both extractive and compressive summarization. These results imply that further improvement of the performance of extractive summarization is possible.

On the other hand, the ROUGE_n scores of the oracle summaries are far from ROUGE_n = 1. We believe that the results are related to the summary’s compression rate. The data set’s compression rate was only 1 to 2%. Thus, under tight length constraints, extractive summarization basically fails to cover large numbers of n-grams in the reference summary. This reveals the limitation of the extractive summarization paradigm and suggests that we need another direction, compressive or abstractive summarization, to overcome the limitation.

6.2.2 ROUGE Scores of Summaries Obtained from Greedy Algorithm

Table 2 also shows the ROUGE_{1,2} scores of the summaries obtained from the greedy algorithm (greedy summaries). Although there are statistically significant differences between the ROUGE

Table 4: Jaccard Index between both oracle and greedy summaries

scores of the oracle summaries and greedy summaries, those obtained from the greedy summaries achieved near optimal scores, *i.e.*, approximation ratio of them are close to 0.9. These results are surprising since the algorithm’s theoretical lower bound is $\frac{1}{2}(1 - \frac{1}{e})(\simeq 0.32)$ OPT.

On the other hand, the results do not support that the differences between them are small at the sentence-level. Table 4 shows the average Jaccard Index between the oracle summaries and the corresponding greedy summaries for the DUC-2004 corpus. The results demonstrate that the oracle summaries are much less similar to the greedy summaries at the sentence-level. Thus, it might not be appropriate to use greedy summaries as training data for learning-based extractive summarization systems.

6.2.3 Impact of Enumeration

Table 5 shows the median number of oracle summaries and the rates of the reference summaries that have multiple oracle summaries for each data set. Over 80% of the reference summaries and about 60% to 90% of the topics have multiple oracle summaries. Since the ROUGE_n scores are based on the unweighted counting of n-grams, when many sentences have similar meanings, *i.e.*, many redundant sentences, the number of oracle summaries that have the same ROUGE_n scores increases. The source documents of multiple document summarization tasks are prone to have many such redundant sentences, and the amount of oracle summaries is large.

	Median				Rate			
	single		multi		single		multi	
	ROUGE ₁	ROUGE ₂	ROUGE ₁	ROUGE ₂	ROUGE ₁	ROUGE ₂	ROUGE ₁	ROUGE ₂
01	8	9	4	5	.854	.787	.833	.733
02	7.5	5.5	4	4	.897	.836	.814	.780
03	8	10.5	3.5	4	.833	.858	.800	.900
04	8	8	3.5	3	.865	.865	.780	.760
05	35	35.5	2	3	.916	.907	.580	.660
06	28	22	2.5	3	.877	.880	.700	.720
07	23	16	4	2	.910	.878	.733	.711

Table 5: Median number of oracle summaries and rates of reference summaries and topics with multiple oracle summaries for each data set

The oracle summaries offer significant benefit with respect to evaluating the extracted sentences. Since both the oracle and system summaries are sets of sentences, it is easy to check whether each sentence in the system summary is contained in one of the oracle summaries. Thus, we can exploit the F-measures, which are useful for evaluating classification-based extractive summarization (Mani and Bloedorn, 1998; Osborne, 2002; Hirao et al., 2002). Here, we have to consider that the oracle summaries, obtained from a reference summary or a set of reference summaries, are not identical at the sentence-level (e.g., the average Jaccard Index between the oracle summaries for the DUC-2004 corpus is around 0.5). The F-measures are varied with the oracle summaries that are used for such computation. For example, assume that we have system summary $S = \{s_1, s_2, s_3, s_4\}$ and oracle summaries $O_1 = \{s_1, s_2, s_5, s_6\}$ and $O_2 = \{s_1, s_2, s_3\}$. The precision for O_1 is 0.5, while that for O_2 is 0.75; the recall for O_1 is 0.5, while that for O_2 is 1; the F-measure for O_1 is 0.5, while that for O_2 is 0.86.

Thus, we employ the scores gained by averaging all of the oracle summaries as evaluation measures. Precision, recall, and F-measure are defined as follows: $P = \{\sum_{O \in O_{all}} |O \cap S| / |S|\} / |O_{all}|$, $R = \{\sum_{O \in O_{all}} |O \cap S| / |O|\} / |O_{all}|$, $F\text{-measure} = 2PR / (P + R)$.

To demonstrate F-measure’s effectiveness, we investigated the correlation between an F-measure and human judgment based on the evaluation results obtained from the DUC-2004 corpus. The results include summaries generated by 17 systems, each of which has a mean coverage score assigned by a human subject. We computed the correla-

tion coefficients between the average F-measure and the average mean coverage score for 50 topics. Table 6 shows Pearson’s r and Spearman’s ρ . In the table, “F-measure (R_1)” and “F-measure (R_2)” indicate the F-measures calculated using oracle summaries optimized to ROUGE₁ and ROUGE₂, respectively. “M” indicates the F-measure calculated using multiple oracle summaries, and “S” indicates F-measures calculated using randomly selected oracle summaries. “multi” indicates oracle summaries obtained from a set of references, and “single” indicates oracle summaries obtained from a reference summary in the set. For “S,” we randomly selected a single oracle summary and calculated the F-measure 100 times and took the average value with the 95% confidence interval of the F-measures by bootstrap resampling.

The results demonstrate that the F-measures are strongly correlated with human judgment. Their values are comparable with those of ROUGE_{1,2}. In particular, F-measure (R_1) (single-M) achieved the best Spearman’s ρ result. When comparing “single” with “multi,” Pearson’s r of “multi” was slightly lower than that of “single,” and the Spearman’s r of “multi” was almost the same as those of “single.” “M” has significantly better performance than “S.” These results imply that F-measures based on oracle summaries are a good evaluation measure and that oracle summaries have the potential to be an alternative to human-made reference summaries in terms of automatic evaluation. Moreover, the enumeration of the oracle summaries for a given reference summary or a set of reference summaries is essential for automatic evaluation.

Metric	r	ρ
ROUGE ₁	.861	.760
ROUGE ₂	.907	.831
F-measure (R ₁) (single-M)	.857	.855
F-measure (R ₁) (single-S)	.815-.830	.811-.830
F-measure (R ₂) (single-M)	.904	.826
F-measure (R ₂) (single-S)	.855-.865	.740-.760
F-measure (R ₁) (multi-M)	.814	.841
F-measure (R ₁) (multi-S)	.794-.802	.803-.813
F-measure (R ₂) (multi-M)	.824	.846
F-measure (R ₂) (multi-S)	.806-.816	.797-.817

Table 6: Correlation coefficients between automatic evaluations and human judgments on DUC-2004 corpus

6.2.4 Search Efficiency

To demonstrate the efficiency of our search algorithm against the naive exhaustive search method, we compared the number of feasible solutions (sets of sentences that satisfy the length constraint) with the number of summaries that were checked in our search algorithm.

Table 7 shows the median number of feasible solutions and checked summaries yielded by our method for each data set (in the case of “single”). The differences in the number of feasible solutions between ROUGE₁ and ROUGE₂ are very large. Input set ($|D|$) of ROUGE₁ is much larger than ROUGE₂. On the other hand, the differences between ROUGE₁ and ROUGE₂ in our method are of the order of 10 to 10². When comparing our method with naive exhaustive searches, its search space is significantly smaller. The differences are of the order of 10⁷ to 10³⁰ with ROUGE₁ and 10⁴ to 10¹⁷ with ROUGE₂. These results demonstrate the efficiency of our branch and bound technique.

In addition, we show an example of the processing time for extracting one oracle summary and enumerating all of the oracle summaries for the reference summaries in the DUC-2004 corpus with a Linux machine (CPU: Intel[®] Xeon[®] X5675 (3.07GHz)) with 192 GB of RAM. We utilized CPLEX 12.1 to solve the ILP problem. Our algorithm was implemented in C++ and compiled with GCC version 4.4.7. The results show that we needed 0.026 and 0.021 sec. to extract one oracle summary per reference summary and 0.047 and 0.031 sec. to extract one oracle summary per set of reference summaries for ROUGE₁ and ROUGE₂, respectively. We needed 11.90 and 1.40 sec. to enumerate the oracle summaries per reference summary and 102.94 and 3.65 sec. per set of reference summaries for ROUGE₁ and

	ROUGE ₁		ROUGE ₂	
	Naive	Proposed	Naive	Proposed
01	3.66×10^{13}	5.75×10^3	3.32×10^7	1.00×10^3
02	1.12×10^{12}	4.64×10^3	1.34×10^7	8.87×10^2
03	1.62×10^{11}	3.65×10^3	6.37×10^6	8.19×10^2
04	9.65×10^{10}	4.47×10^3	6.90×10^6	9.83×10^2
05	5.48×10^{36}	2.32×10^6	3.48×10^{21}	7.03×10^4
06	1.94×10^{32}	1.97×10^6	2.11×10^{20}	5.08×10^4
07	4.14×10^{28}	1.40×10^6	1.81×10^{19}	2.60×10^4

Table 7: Median number of summaries checked by each search method

ROUGE₂, respectively. The extraction of one oracle summary for a reference summary can be achieved with the ILP solver in practical time and the enumeration of oracle summaries is also efficient. However, to enumerate oracle summaries, we needed several weeks for some topics in DUCs 2005 to 2007 since they hold a huge number of source sentences.

7 Conclusions

To analyze the limitations and the future direction of extractive summarization, this paper proposed (1) Integer Linear Programming (ILP) formulation to obtain extractive oracle summaries in terms of ROUGE_n scores and (2) an algorithm that enumerates all oracle summaries to exploit F-measures that evaluate the sentences extracted by systems.

The evaluation results obtained from the corpora of DUCs 2001 to 2007 identified the following: (1) room still exists to improve the ROUGE_n scores of extractive summarization systems even though the ROUGE_n scores of the oracle summaries fell below the theoretical upper bound ROUGE_n=1. (2) Over 80% of the reference summaries and from 60% to 90% of the sets of reference summaries have multiple oracle summaries, and the F-measures computed by utilizing the enumerated oracle summaries showed stronger correlation with human judgment than those computed from single oracle summaries.

Acknowledgments

The authors thank three anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

- Miguel B. Almeida and André F.T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 196–206.
- Soddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ILP based multi-sentence compression. In *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1208–1214.
- Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1587–1597.
- Hakan Ceylan, Rada Mihalcea, Umut Özertem, Elena Lloret, and Manuel Palomar. 2010. Quantifying the limits and success of extractive summarization systems across domains. In *Proc. of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 903–911.
- John M. Conroy, Jade Goldstein, Judith D. Schlesinger, and Dianne P. O’Leary. 2004. Left-brain/right-brain multi-document summarization. In *Proc. of the Document Understanding Conference (DUC)*.
- John M. Conroy, Judith D. Schlesinger, Jeff Kubina, Peter A. Rankel, and Dianne P. O’Leary. 2011. Classy 2011 at TAC: Guided and multi-lingual summaries and evaluation metrics. In *Proc. of the Text Analysis Conference (TAC)*.
- John M. Conroy, Sashka T. Davis, Jeff Kubina, Yi-Kai Liu, Dianne P. O’Leary, and Judith D Schlesinger. 2013. Multilingual summarization: Dimensionality reduction and a step towards optimal term coverage. In *Proc. of the MultiLing 2013 Workshop on Multilingual Multi-document Summarization*, pages 55–63.
- Sashka T. Davie, John M. Conroy, and Judith D. Schlesinger. 2012. OCCAMS - an optimal combinatorial covering algorithm for multi-document summarization. In *Proc. of the 12th IEEE International Conference on Data Mining Workshops, ICDM Workshops*, pages 454–463.
- Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proc. of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18.
- Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD summarization system at TAC 2009. In *Proc. of the Text Analysis Conference (TAC)*.
- Tsutomu Hirao, Hideki Isozaki, Eisaku Maeda, and Yuji Matsumoto. 2002. Extracting import sentences with support vector machines. In *Proc. of the 19th International Conference on Computational Linguistics (COLING)*, pages 342–348.
- Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721.
- Kai Hong, John Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proc. of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, pages 1608–1616.
- Kai Hong, Mitchell Marcus, and Ani Nenkova. 2015. System combination for multi-document summarization. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 107–117.
- Samir Khuller, Anna Moss, and Joseph Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.
- Alex Kulesza and Ben Tasker. 2011. Learning determinantal point process. In *Proc. of the 27th Conference on Uncertainty in Artificial Intelligence*.
- Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proc. of the 49th Association for Computational Linguistics: Human Language Technologies*, pages 510–520.
- Hui Lin and Jeff Bilmes. 2012. Learning mixtures of submodular shells with application to document summarization. In *Proc. of the 28th Conference on Uncertainty in Artificial Intelligence (UAI2012)*.
- Chin-Yew Lin and Eduard Hovy. 2003. The potential and limitations of automatic sentence extraction for summarization. In *Proc. of the HLT-NAACL 03 Text Summarization Workshop*, pages 73–80.
- Inderjeet Mani and Eric Bloedorn. 1998. Machine learning of generic and user-focused summarization. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 820–826.
- Hans Moen, Juho Heimonen, Laura-Maria Murtola, Antti Airola, Tapio Pahikkala, Virpi Terv, Riitta Danielsson-Ojala, Tapio Salakoski, and Sanna Salanter. 2014. On evaluation of automatically generated clinical discharge summaries. In *Proc. of the 2nd European Workshop on Practical Aspects of Health Informatics*, pages 101–114.
- Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 1–8.

- Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proc. of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, June.
- Daraksha Parveen, Hans-Martin Ramsel, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954, Lisbon, Portugal, September. Association for Computational Linguistics.
- Maxime Peyrard and Judith Eckle-Köhler. 2016. Optimizing an approximation of rouge - a problem-reduction approach to extractive multi-document summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1836, Berlin, Germany, August. Association for Computational Linguistics.
- Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.
- Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502.
- Korbinian Riedhammer, Dan Gillick, Benoit Favre, and Dilek Hakkani-Tür. 2008. Packing the meeting summarization knapsack. In *Proc. of the 9th Annual Conference of the International Speech Communication Association*, pages 2434–2437.
- Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin learning of submodular summarization models. In *Proc. of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233.
- Hiroya Takamura and Manabu Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proc. of the 12th Conference of the European of the Association for Computational Linguistics*, pages 781–789.
- Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Compressive document summarization via sparse optimization. In *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1376–1382.
- Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1961–1966, Lisbon, Portugal, September. Association for Computational Linguistics.

Neural Semantic Encoders

Tsendsuren Munkhdalai and Hong Yu

University of Massachusetts, MA, USA

tsendsuren.munkhdalai@umassmed.edu, hong.yu@umassmed.edu

Abstract

We present a memory augmented neural network for natural language understanding: Neural Semantic Encoders. NSE is equipped with a novel memory update rule and has a variable sized encoding memory that evolves over time and maintains the understanding of input sequences through *read*, *compose* and *write* operations. NSE can also access¹ multiple and shared memories. In this paper, we demonstrated the effectiveness and the flexibility of NSE on five different natural language tasks: natural language inference, question answering, sentence classification, document sentiment analysis and machine translation where NSE achieved state-of-the-art performance when evaluated on publically available benchmarks. For example, our shared-memory model showed an encouraging result on neural machine translation, improving an attention-based baseline by approximately 1.0 BLEU.

1 Introduction

Recurrent neural networks (RNNs) have been successful for modeling sequences (Elman, 1990). Particularly, RNNs equipped with internal short memories, such as long short-term memories (LSTM) (Hochreiter and Schmidhuber, 1997) have achieved a notable success in sequential tasks (Cho et al., 2014; Vinyals et al., 2015). LSTM is powerful because it learns to control its short term memories. However, the short term memories in LSTM are a part of the training parameters. This imposes some practical difficulties in training and modeling long sequences with LSTM.

¹By access we mean changing the memory states by the *read*, *compose* and *write* operations.

Recently several studies have explored ways of extending the neural networks with an external memory (Graves et al., 2014; Weston et al., 2015; Grefenstette et al., 2015). Unlike LSTM, the short term memories and the training parameters of such a neural network are no longer coupled and can be adapted. In this paper we propose a novel class of memory augmented neural networks called Neural Semantic Encoders (NSE) for natural language understanding. NSE offers several desirable properties. NSE has a variable sized encoding memory which allows the model to access entire input sequence during the reading process; therefore efficiently delivering long-term dependencies over time. The encoding memory evolves over time and maintains the memory of the input sequence through *read*, *compose* and *write* operations. NSE sequentially processes the input and supports word compositionality inheriting both temporal and hierarchical nature of human language. NSE can read from and write to a set of relevant encoding memories simultaneously or multiple NSEs can access a shared encoding memory effectively supporting knowledge and representation sharing. NSE is flexible, robust and suitable for practical NLU tasks and can be trained easily by any gradient descent optimizer.

We evaluate NSE on five different real tasks. For four of them, our models set new state-of-the-art results. Our results suggest that a NN model with the shared memory between encoder and decoder is a promising approach for sequence transduction problems such as machine translation and abstractive summarization. In particular, we observe that the attention-based neural machine translation can be further improved by shared-memory models. We also analyze memory access pattern and compositionality in NSE and show that our model captures semantic and syntactic structures of input sentence.

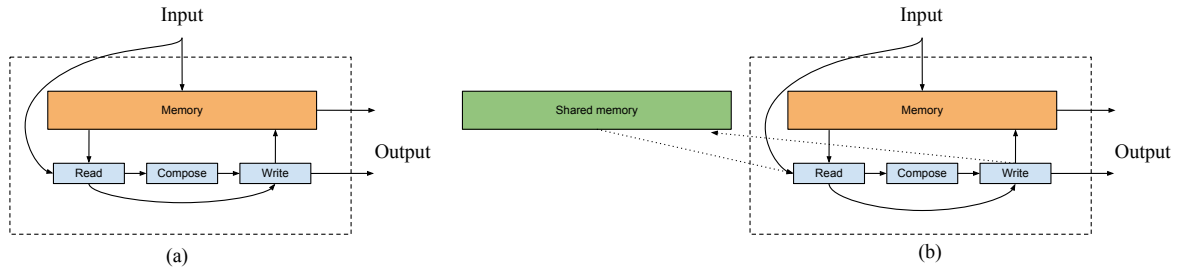


Figure 1: High-level architectures of the Neural Semantic Encoders. NSE reads and writes its own encoding memory in each time step (a). MMA-NSE accesses multiple relevant memories simultaneously (b).

2 Related Work

One of the pioneering work that attempts to extend deep neural networks with an external memory is Neural Turing Machines (NTM) (Graves et al., 2014). NTM implements a centralized controller and a fixed-sized random access memory. The NTM memory is addressable by both content (i.e. soft attention) and location based access mechanisms. The authors evaluated NTM on algorithmic tasks such as copying and sorting sequences.

Comparison with Neural Turing Machines: NSE addresses certain drawbacks of NTM. NTM has a single centralized controller, which is usually an MLP or RNN while NSE takes a modular approach. The main controller in NSE is decomposed into three separate modules, each of which performs for *read*, *compose* or *write* operation. In NSE, the *compose* module is introduced in addition to the standard memory update operations (i.e. read-write) in order to process the memory entries and input information.

The main advantage of NSE over NTM is in its memory update. Despite its sophisticated addressing mechanism, the NTM controller does not have mechanism to avoid *information collision* in the memory. Particularly the NTM controller emits two separate set of access weights (i.e. read weight and erase and write weights) that do not explicitly encode the knowledge about where information is read from and written to. Moreover the fixed-size memory in NTM has no memory allocation or de-allocation protocol. Therefore unless the controller is intelligent enough to track the previous read/write information, which is hard for an RNN when processing long sequences, the memory content is overlapped and information is over-

written throughout different time scales. We think that this is a potential reason that makes NTM hard to train and makes the training not stable. We also note that the effectiveness of the location based addressing introduced in NTM is unclear. In NSE, we introduce a novel and systematic memory update approach based on the soft attention mechanism. NSE writes new information to the most recently read memory locations. This is accomplished by sharing the same memory key vector between the read and write modules. The NSE memory update is scalable and potentially more robust to train. NSE is provided with a variable sized memory and thus unlike NTM, the size of the NSE memory is more relaxed. The novel memory update mechanism and the variable sized memory together prevent NSE from the *information collision* issue and avoid the need of the memory allocation and de-allocation protocols. Each memory location of the NSE memory stores a token representation in input sequence during encoding. This provides NSE with an anytime-access to the entire input sequence including the tokens from the future time scales, which is not permitted in NTM, RNN and attention-based encoders.

Lastly, NTM addresses small algorithmic problems while NSE focuses on a set of large-scale language understanding tasks.

The RNNSearch model proposed in (Bahdanau et al., 2015) can be seen as a variation of memory augmented networks due to its ability to read the historic output states of RNNs with soft attention. The work of Sukhbaatar et al. (2015) combines the soft attention with Memory Networks (MemNNs) (Weston et al., 2015). Similar to RNNSearch, MemNNs are designed with non-writable memories. It constructs layered memory representa-

tions and showed promising results on both artificial and real question answering tasks. We note that RNNSearch and MemNNs avoid the memory update and management overhead by simply using a non-writable memory storage. Another variation of MemNNs is Dynamic Memory Network (Kumar et al., 2016) that is equipped with an episodic memory and seems to be flexible in different settings.

Although NSE differs from other memory-augmented NN models in many aspects, they all use soft attention mechanism with a type of similarity measures to retrieve relevant information from the external memory. For example, NTM implements cosine similarity and MemNNs use vector dot product. NSE uses the vector dot product for the similarity measure in NSE because it is faster to compute.

Other related work includes Neural Program-Interpreters (Reed and de Freitas, 2016), which learns to run sub-programs and to compose them for high-level programs. It uses execution traces to provide the full supervision. Researchers have also explored ways to add unbounded memory to LSTM (Grefenstette et al., 2015) using a particular data structure. Although this type of architecture provides a flexible capacity to store information, the memory access is constrained by the data structure used for the memory bank, such as stack and queue.

Overall it is expensive to train and to scale the previously proposed memory-based models. Most models required a set of clever engineering tricks to work successfully. Most of the aforementioned memory augmented neural networks have been tested on synthetic tasks whereas in this paper we evaluated NSE on a wide range of real and large-scale natural language applications.

3 Proposed Approach

Our training set consists of N examples $\{X^i, Y^i\}_{i=1}^N$, where the input X^i is a sequence $w_1^i, w_2^i, \dots, w_{T_i}^i$ of tokens while the output Y^i can be either a single target or a sequence. We transform each input token w_t to its word embedding x_t .

Our Neural Semantic Encoders (NSE) model has four main components: read, compose and write modules and an encoding memory $M \in R^{k \times l}$ with a variable number of slots, where k is the embedding dimension and l is the length

of the input sequence. Each memory slot vector $m_t \in R^k$ corresponds to the vector representation of information about word w_t in memory. In particular, the memory is initialized by the embedding vectors $\{x_t\}_{t=1}^l$ and is evolved over time, through *read*, *compose* and *write* operations. Figure 1 (a) illustrates the architecture of NSE.

3.1 Read, Compose and Write

NSE performs three main operations in every time step. After initializing the memory slots with the corresponding input representations, NSE processes an embedding vector x_t and retrieves a memory slot $m_{r,t}$ that is expected to be associatively coherent (i.e. semantically associated) with the current input word w_t .² The slot location r (ranging from 1 to l) is defined by a key vector z_t which the read module emits by attending over the memory slots. The compose module implements a composition operation that combines the memory slot with the current input. The write module then transforms the composition output to the encoding memory space and writes the resulting new representation into the slot location of the memory. Instead of composing the raw embedding vector x_t , we use the hidden state o_t produced by the read module at time t

Concretely, let $e_l \in R^l$ and $e_k \in R^k$ be vectors of ones and given a read function f_r^{LSTM} , a composition f_c^{MLP} and a write f_w^{LSTM} NSE in Figure 1 (a) computes the key vector z_t , the output state h_t , and the encoding memory M_t in time step t as

$$o_t = f_r^{LSTM}(x_t) \quad (1)$$

$$z_t = \text{softmax}(o_t^\top M_{t-1}) \quad (2)$$

$$m_{r,t} = z_t^\top M_{t-1} \quad (3)$$

$$c_t = f_c^{MLP}(o_t, m_{r,t}) \quad (4)$$

$$h_t = f_w^{LSTM}(c_t) \quad (5)$$

$$M_t = M_{t-1}(\mathbf{1} - (z_t \otimes e_k)^\top) + (h_t \otimes e_l)(z_t \otimes e_k)^\top \quad (6)$$

where $\mathbf{1}$ is a matrix of ones, \otimes denotes the outer product which duplicates its left vector l or k times to form a matrix. The read function f_r^{LSTM} sequentially maps the word embeddings to the internal space of the memory M_{t-1} . Then Equation 2 looks for the slots related to the input by computing association degree between each memory slot

²Such a coherence is calculated by a soft attention with dot product similarity.

and the hidden state o_t . We calculate the association degree by the dot product and transform this scores to the fuzzy key vector z_t by normalizing with *softmax* function. Since our key vector is fuzzy, the slot to be composed is retrieved by taking weighted sum of the all slots as in Equation 3. This process can also be seen as the soft attention mechanism (Bahdanau et al., 2015). In Equation 4 and 5, we compose and process the retrieved slot with the current hidden state and map the resulting vector to the encoder output space. Finally, we write the new representation to the memory location pointed by the key vector in Equation 6 where the key vector z_t emitted by the read module is reused to inform the write module of the most recently read slots. First the slot information that was retrieved is erased and then the new representation is located. NSE performs this iterative process until all words in the input sequence are read. The encoding memories $\{M\}_{t=1}^T$ and output states $\{h\}_{t=1}^T$ are further used for the tasks.

Although NSE reads a single word at a time, it has an anytime-access to the entire sequence stored in the encoding memory. With the encoding memory, NSE maintains a mental image of the input sequence. The memory is initialized with the raw embedding vector at time $t = 0$. We term such a freshly initialized memory a *baby* memory. As NSE reads more input content in time, the *baby* memory evolves and refines the encoded mental image.

The read f_r^{LSTM} , the composition f_c^{MLP} and the write f_w^{LSTM} functions are neural networks and are the training parameters in our NSE. As the name suggests, we use LSTM and multi-layer perceptron (MLP) in this paper. Since NSE is fully differentiable, it can be trained with any gradient descent optimizer.

3.2 Shared and Multiple Memory Accesses

For sequence to sequence transduction tasks like question answering, natural language inference and machine translation, it is beneficial to access other relevant memories in addition to its own one. The shared or the multiple memory access allows a set of NSEs to exchange knowledge representations and to communicate with each other to accomplish a particular task throughout the encoding memory.

NSE can be extended easily, so that it is able to read from and write to multiple memories si-

multaneously or multiple NSEs are able to access a shared memory. Figure 1 (b) depicts a high-level architectural diagram of a multiple memory access-NSE (MMA-NSE). The first memory (in green) is the shared memory accessed by more than one NSEs. Given a shared memory $M^n \in R^{k \times n}$ that has been encoded by processing a relevant sequence with length n , MMA-NSE with the access to one relevant memory is defined as

$$o_t = f_r^{LSTM}(x_t) \quad (7)$$

$$z_t = softmax(o_t^\top M_{t-1}) \quad (8)$$

$$m_{r,t} = z_t^\top M_{t-1} \quad (9)$$

$$z_t^n = softmax(o_t^\top M_{t-1}^n) \quad (10)$$

$$m_{r,t}^n = z_t^{n\top} M_{t-1}^n \quad (11)$$

$$c_t = f_c^{MLP}(o_t, m_{r,t}, m_{r,t}^n) \quad (12)$$

$$h_t = f_w^{LSTM}(c_t) \quad (13)$$

$$M_t = M_{t-1}(\mathbf{1} - (z_t \otimes e_k)^\top) + (h_t \otimes e_l)(z_t \otimes e_k)^\top \quad (14)$$

$$M_t^n = M_{t-1}^n(\mathbf{1} - (z_t^n \otimes e_k)^\top) + (h_t \otimes e_n)(z_t^n \otimes e_k)^\top \quad (15)$$

and this is almost the same as standard NSE. The read module now emits the additional key vector z_t^n for the shared memory and the composition function f_c^{MLP} combines more than one slots.

In MMA-NSE, the different memory slots are retrieved from the shared memories depending on their encoded semantic representations. They are then composed together with the current input and written back to their corresponding slots. Note that MMA-NSE is capable of accessing a variable number of relevant shared memories once a composition function that takes in dynamic inputs is chosen.

4 Experiments

We describe in this section experiments on five different tasks, in order to show that NSE can be effective and flexible in different settings.³ We report results on natural language inference, question answering (QA), sentence classification, document sentiment analysis and machine translation. All five tasks challenge a model in terms of language understanding and semantic reasoning.

The models are trained using Adam (Kingma and Ba, 2014) with hyperparameters selected on

³Code for the experiments and NSEs is available at <https://bitbucket.org/tsendeemts/nse>.

Model	d	$ \theta _M$	Train	Test
Classifier with handcrafted features (Bowman et al., 2015)	-	-	99.7	78.2
LSTM encoders (Bowman et al., 2015)	300	3.0M	83.9	80.6
Dependency Tree CNN encoders (Mou et al., 2016)	300	3.5M	83.3	82.1
SPINN-PI encoders (Bowman et al., 2016)	300	3.7M	89.2	83.2
NSE	300	3.4M	86.2	84.6
MMA-NSE	300	6.3M	87.1	84.8
LSTM attention (Rocktäschel et al., 2016)	100	242K	85.4	82.3
LSTM word-by-word attention (Rocktäschel et al., 2016)	100	252K	85.3	83.5
MMA-NSE attention	300	6.5M	86.9	85.4
mLSTM word-by-word attention (Wang and Jiang, 2015)	300	1.9M	92.0	86.1
LSTMN with deep attention fusion (Cheng et al., 2016)	450	3.4M	89.5	86.3
Decomposable attention model (Parikh et al., 2016)	200	582K	90.5	86.8
Full tree matching NTI-SLSTM-LSTM global attention (Munkhdalai and Yu, 2017)	300	3.2M	88.5	87.3

Table 1: Training and test accuracy on natural language inference task. d is the word embedding size and $|\theta|_M$ the number of model parameters.

development set. We chose two one-layer LSTM for read/write modules on the tasks other than QA on which we used two-layer LSTM. The pre-trained 300-D Glove 840B vectors and 100-D Glove 6B vectors (Pennington et al., 2014) were obtained for the word embeddings.⁴ The word embeddings are fixed during training. The embeddings for out-of-vocabulary words were set to zero vector. We crop or pad the input sequence to a fixed length. A padding vector was inserted when padding. The models were regularized by using dropouts and an l_2 weight decay.⁵

4.1 Natural Language Inference

The natural language inference is one of the main tasks in language understanding. This task tests the ability of a model to reason about the semantic relationship between two sentences. In order to perform well on the task, NSE should be able to capture sentence semantics and be able to reason the relation between a sentence pair, i.e., whether a premise-hypothesis pair is entailing, contradictory or neutral. We conducted experiments on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), which consists of 549,367/9,842/9,824 premise-hypothesis pairs for train/dev/test sets and target label indicating their relation.

Following the setting in (Mou et al., 2016; Bowman et al., 2016) the NSE output for each sentence was the input to a MLP, where the input layer computes the concatenation $[h_i^p; h_i^h]$, absolute difference $h_i^p - h_i^h$ and elementwise product $h_i^p \cdot h_i^h$ of the two sentence representations. In addition, the MLP has a hidden layer with 1024 units with

ReLU activation and a *softmax* layer. We set the batch size to 128, the initial learning rate to $3e-4$ and l_2 regularizer strength to $3e-5$, and train each model for 40 epochs. The write/read neural nets and the last linear layer were regularized by using 30% dropouts.

We evaluated three different variations of NSE show in Table 1. The NSE model encodes each sentence simultaneously by using a separate memory for each sentence. The second model - MMA-NSE first encodes the premise and then the hypothesis sentence by sharing the premise encoded memory in addition to the hypothesis memory. For the third model, we use inter-sentence attention which selectively reconstructs the premise representation.

Table 1 shows the results of our models along with the results of published methods for the task. The classifier with handcrafted features extracts a set of lexical features. The next group of models are based on sentence encoding. While most of the sentence encoder models rely solely on word embeddings, the dependency tree CNN and the SPINN-PI models make use of sentence parser output. The SPINN-PI model is similar to NSE in spirit that it also explicitly computes word composition. However, the composition in the SPINN-PI is guided by supervisions from a dependency parser. NSE outperformed the previous sentence encoders on this task. The MMA-SNE further slightly improved the result, indicating that reading the premise memory is helpful while encoding the hypothesis.

The last set of methods designs inter-sentence relation with parameterized soft attention (Bahdanau et al., 2015). Our MMA-NSE attention model is similar to the LSTM attention model.

⁴<http://nlp.stanford.edu/projects/glove/>

⁵More detail on hyper-parameters can be found in code.

Particularly, it attends over the premise encoder outputs $\{h^p\}_{t=1}^T$ in respect to the final hypothesis representation h_t^h and constructs an attentively blended vector of the premise. This model obtained 85.4% accuracy score. The best performing model for this task performs tree matching with attention mechanism and LSTM.

4.2 Answer Sentence Selection

Answer sentence selection is an integral part of the open-domain question answering. For this task, a model is trained to identify the correct sentences that answer a factual question, from a set of candidate sentences. We experiment on WikiQA dataset constructed from Wikipedia (Yang et al., 2015). The dataset contains 20,360/2,733/6,165 QA pairs for train/dev/test sets.

The MLP setup used in the language inference task is kept same, except that we now replace the *softmax* layer with a *sigmoid* layer and model the following conditional probability distribution.

$$p_{\theta}(y = 1|h_t^q, h_t^a) = \text{sigmoid}(o^{QA}) \quad (16)$$

where h_t^q and h_t^a are the question and the answer encoded vectors and o^{QA} denotes the output of the hidden layer of the MLP. We trained the MMA-NSE attention model to minimize the *sigmoid* cross entropy loss. MMA-NSE first encodes the answers and then the questions by accessing its own and the answer encoding memories. In our preliminary experiment, we found that the multiple memory access and the attention over answer encoder outputs $\{h^a\}_{t=1}^T$ are crucial to this problem. Following previous work, we adopt MAP and MRR as the evaluation metrics for this task.⁶

We set the batch size to 4 and the initial learning rate to 1e-5, and train the model for 10 epochs. We used 40% dropouts after word embeddings and no

⁶We used *trac_eval* script to calculate the evaluation metrics

Model	MAP	MRR
Classifier with features (2013)	0.5993	0.6068
Paragraph Vector (2014)	0.5110	0.5160
Bigram-CNN (2014)	0.6190	0.6281
3-layer LSTM (2016)	0.6552	0.6747
3-layer LSTM attention (2016)	0.6639	0.6828
NASM (2016)	0.6705	0.6914
MMA-NSE attention	0.6811	0.6993

Table 2: Experiment results on answer sentence selection.

l_2 weight decay. The word embeddings are pre-trained 300-D Glove 840B vectors. For this task, a linear mapping layer transforms the 300-D word embeddings to the 512-D LSTM inputs.

Table 2 presents the results of our model and the previous models for the task.⁷ The classifier with handcrafted features is a SVM model trained with a set of features. The Bigram-CNN model is a simple convolutional neural net. While the LSTM and LSTM attention models outperform the previous best result by nearly 5-6% by implementing deep LSTM with three hidden layers, NASM improves it further and sets a strong baseline by combining variational auto-encoder (Kingma and Welling, 2014) with the soft attention. Our MMA-NSE attention model exceeds the NASM by approximately 1% on MAP and 0.8% on MRR for this task.

4.3 Sentence Classification

We evaluated NSE on the Stanford Sentiment Treebank (SST) (Socher et al., 2013). This dataset comes with standard train/dev/test sets and two subtasks: binary sentence classification or fine-grained classification of five classes. We trained our model on the text spans corresponding to labeled phrases in the training set and evaluated the model on the full sentences.

The sentence representations were passed to a two-layer MLP for classification. The first layer of the MLP has *ReLU* activation and 1024 or 300 units for binary or fine-grained setting. The second layer is a *softmax* layer. The read/write modules are two one-layer LSTM with 300 hidden units and the word embeddings are the pre-trained 300-D Glove 840B vectors. We set the batch size to 64, the initial learning rate to 3e-4 and l_2 regularizer strength to 3e-5, and train each model for 25 epochs. The write/read neural nets and the last linear layer were regularized by 50% dropouts.

Table 3 compares the result of our model with the state-of-the-art methods on the two subtasks. Most best performing methods exploited the parse tree provided in the treebank on this task with the exception of the DMN. The Dynamic Memory Network (DMN) model is a memory-augmented network. Our model outperformed the DMN and set the state-of-the-art results on both subtasks.

⁷Inclusion of simple word count feature improves the performance by around 0.15-0.3 across the board

Model	Bin	FG
RNTN (Socher et al., 2013)	85.4	45.7
Paragraph Vector (Le and Mikolov, 2014)	87.8	48.7
CNN-MC (Kim, 2014)	88.1	47.4
DRNN (Irsoy and Cardie, 2015)	86.6	49.8
2-layer LSTM(Tai et al., 2015)	86.3	46.0
Bi-LSTM(Tai et al., 2015)	87.5	49.1
CT-LSTM(Tai et al., 2015)	88.0	51.0
DMN (Kumar et al., 2016)	88.6	52.1
NSE	89.7	52.8

Table 3: Test accuracy for sentence classification. Bin: Binary, FG: fine-grained 5 classes.

4.4 Document Sentiment Analysis

We evaluated our models for document-level sentiment analysis on two publically available large-scale datasets: the IMDB consisting of 335,018 movie reviews and 10 different classes and Yelp 13 consisting of 348,415 restaurant reviews and 5 different classes. Each document in the datasets is associated with human ratings and we used these ratings as gold labels for sentiment classification. Particularly, we used the pre-split datasets of (Tang et al., 2015).

We stack a NSE or LSTM on the top of another NSE for document modeling. The first NSE encodes the sentences and the second NSE or LSTM takes sentence encoded outputs and constructs document representations. The document representation is given to a output *softmax* layer. The whole network is trained jointly by backpropagating the cross entropy loss. We used one-layer LSTM with 100 hidden units for the read/write modules and the pre-trained 100-D Glove 6B vectors for this task. We set the batch size to 32, the initial learning rate to $3e-4$ and l_2 regularizer strength to $1e-5$, and trained each model for 50 epochs. The write/read neural nets and the document-level NSE/LSTM were regularized by 15% dropouts and the softmax layer by 20% dropouts. In order to speedup the training, we created document buckets by considering the number of sentences per document, i.e., documents with the same number of sentences were put together in the same bucket. The buckets were shuffled and updated per epoch. We did not use curriculum scheduling (Bengio et al., 2009), although it is observed to help sequence training.

Table 4 shows our results. We report two performance metrics: accuracy and MSE. The best results on the task were previously obtained by Conv-GRNN and LSTM-GRNN, which are also

Model	Yelp 13		IMDB	
	Acc	MSE	Acc	MSE
Classifier (2015)	59.8	0.68	40.5	3.56
PV (2015)	57.7	0.86	34.1	4.69
CNN (2015)	59.7	0.76	37.6	3.30
Conv-GRNN (2015)	63.7	0.56	42.5	2.71
LSTM-GRNN (2015)	65.1	0.50	45.3	3.00
NSE-NSE	66.6	0.48	48.3	1.94
NSE-LSTM	67.0	0.47	48.1	1.98

Table 4: Results of document-level sentiment classification. PV: paragraph vector, Acc: accuracy, and MSE: mean squared error.

Model	Train	Dev	Test
Baseline LSTM-LSTM	28.06	17.96	17.02
NSE-LSTM	28.73	17.67	17.13
NSE-NSE	29.89	18.53	17.93

Table 5: BLEU scores for English-German translation task.

stacked models. These models first learn the sentence representations with a CNN or LSTM and then combine them for document representation using a gated recurrent neural network (GRNN). Our NSE models outperformed the previous state-of-the-art models in terms of both accuracy and MSE, by approximately 2-3%. On the other hand, all systems tend to show poor results on the IMDB dataset. That is, the IMDB dataset contains longer documents than the Yelp 13 and it has 10 classes while the Yelp 13 dataset has five classes to distinguish.⁸ The stacked NSEs (NSE-NSE) performed slightly better than the NSE-LSTM on the IMDB dataset. This is possibly due to the encoding memory of the document level NSE that preserves the long dependency in documents with a large number of sentences.

4.5 Machine Translation

Lastly, we conducted an experiment on neural machine translation (NMT). The NMT problem is mostly defined within the encoder-decoder framework (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). The encoder provides the semantic and syntactic information about the source sentences to the decoder and the decoder generates the target sentences by conditioning on this information and its partially produced translation. For an efficient encoding, the attention-based NMT was introduced (Bahdanau et al., 2015).

⁸The average number of sentences and words in a document for IMDB: 14, 152 and Yelp 13: 9, 326

For NTM, we implemented three different models. The first model is a baseline model and is similar to the one proposed in (Bahdanau et al., 2015) (RNNSearch). This model (LSTM-LSTM) has two LSTM for the encoder/decoder and has the soft attention neural net, which attends over the source sentence and constructs a focused encoding vector for each target word. The second model is an NSE-LSTM encoder-decoder which encodes the source sentence with NSE and generates the targets with the LSTM network by using the NSE output states and the attention network. The last model is an NSE-NSE setup, where the encoding part is the same as the NSE-LSTM while the decoder NSE now uses the output state and has an access to the encoder memory, i.e., the encoder and the decoder NSEs access a shared memory. The memory is encoded by the first NSEs and then read/written by the decoder NSEs. We used the English-German translation corpus from the IWSLT 2014 evaluation campaign (Cettolo et al., 2012). The corpus consists of sentence-aligned translation of TED talks. The data was pre-processed and lowercased with the Moses toolkit.⁹ We merged the dev2010 and dev2012 sets for development and the tst2010, tst2011 and tst2012 sets for test data¹⁰. Sentence pairs with length longer than 25 words were filtered out. This resulted in 110,439/4,998/4,793 pairs for train/dev/test sets. We kept the most frequent 25,000 words for the German dictionary. The English dictionary has 51,821 words. The 300-D Glove 840B vectors were used for embedding the words in the source sentence whereas a lookup embedding layer was used for the target German words. Note that the word embeddings are usually optimized along with the NMT models. However, for the evaluation purpose we in this experiment do not optimize the English word embeddings. Besides, we do not use a beam search to generate the target sentences.

The LSTM encoder/decoders have two layers with 300 units. The NSE read/write modules are two one-layer LSTM with the same number of units as the LSTM encoder/decoders. This ensures that the number of parameters of the models is roughly the equal. The models were trained to minimize word-level cross entropy loss and were regularized by 20% input dropouts and the

⁹<https://github.com/moses-smt/mosesdecoder>

¹⁰We modified `prepareData.sh` script: <https://github.com/facebookresearch/MIXER>

30% output dropouts. We set the batch size to 128, the initial learning rate to 1e-3 for LSTM-LSTM and 3e-4 for the other models and l_2 regularizer strength to 3e-5, and train each model for 40 epochs. We report BLEU score for each models.¹¹

Table 5 reports our results. The baseline LSTM-LSTM encoder-decoder (with attention) obtained 17.02 BLEU on the test set. The NSE-LSTM improved the baseline slightly. Given this very small improvement of the NSE-LSTM, it is unclear whether the NSE encoder is helpful in NMT. However, if we replace the LSTM decoder with another NSE and introduce the shared memory access to the encoder-decoder model (NSE-NSE), we improve the baseline result by almost 1.0 BLEU. The NSE-NSE model also yields an increasing BLEU score on dev set. The result demonstrates that the attention-based NMT systems can be improved by a shared-memory encoder-decoder model. In addition, memory-based NMT systems should perform well on translation of long sequences by preserving long term dependencies.

5 Qualitative Analysis

5.1 Memory Access and Compositionality

NSE is capable of performing multiscale composition by retrieving associative slots for a particular input at a time step. We analyzed the memory access order and the compositionality of memory slot and the input word in the NSE model trained on the SNLI data.

Figure 2 shows the word association graphs for the two sentence picked from SNLI test set. The association graph was constructed by inspecting the key vector z . For an input word, we connect it to the most active slot pointed by z ¹².

Note the graph components clustered around the semantically rich words: "sits", "wall" and "autumn" (a) and "Three", "puppies", "tub" and "vet" (b). The memory slots corresponding to words that are semantically rich in the current context are the most frequently accessed. The graph is able to capture certain syntactic structures including phrases (e.g., "hand built rock wall") and modifier relations (between "sits" and "quietly" and

¹¹We computed the BLEU score with `multi-bleu.perl` script of the Moses toolkit

¹²Since z is fuzzy, we visualize the highest scoring slot. For a few inputs, z pointed to a slot corresponding to the same word. In this case, we masked out those slots and showed the second best scoring slot.

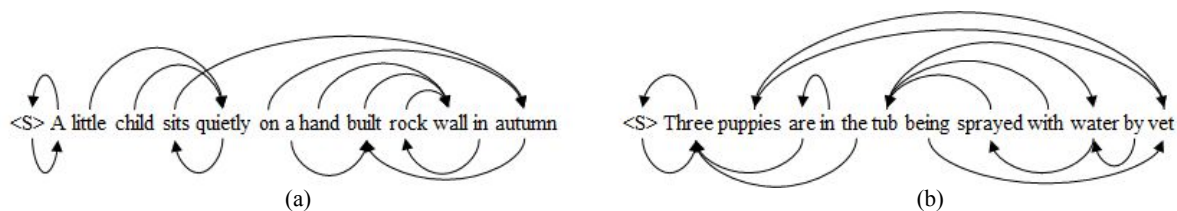


Figure 2: Word association or composition graphs produced by NSE memory access. The directed arcs connect the words that are composed via *compose* module. The source nodes are input words and the destination nodes (pointed by the arrows) correspond to the accessed memory slots. $\langle S \rangle$ denotes the beginning of sequence.

between "tub" and "sprayed with water"). Another interesting property is that the model tends to perform sensible compositions while processing the input sentence. For example, NSE retrieved the memory slot corresponding to "wall" or "Three" when reading the input "rock" or "are".

In Appendix A, we show a step-by-step visualization of NSE memory states for the first sentence. Note how the encoding memory is evolved over time. In time step four ($t = 4$), the memory slot for "quietly" encodes information about "quiet(ly) little child". When $t = 6$, the model forms another composition involving "quietly", "quietly sits". In the last time step, we are able to find the most or the least frequently accessed slots in the memory. The least accessed slots correspond to function words while the frequently accessed slots are content words and tend to carry out rich semantics and intrinsic compositions found in the input sentence. Overall the model is less constrained and is able to compose multiword expressions.

6 Conclusion

Our proposed memory augmented neural networks have achieved the state-of-the-art results when evaluated on five representative NLP tasks. NSE is capable of building an efficient architecture of the single, shared and multiple memory accesses for a specific NLP task. For example, for the NLI task NSE accesses premise encoded memory when processing hypothesis. For the QA task, NSE accesses answer encoded memory when reading question for QA. In machine translation, NSE shares a single encoded memory between encoder and decoder. Such flexibility in the architectural choice of the NSE memory access allows for

the robust models for a better performance.

The initial state of the NSE memory stores information about each word in the input sequence. We in this paper used word embeddings to represent the words in the memory. Different variations of word representations such as character-based models are left to be evaluated for memory initialization in the future. We plan to extend NSE so that it learns to select and access a relevant subset from a memory set. One could also explore unsupervised variations of NSE, for example, to train them to produce encoding memory and representation vector of entire sentences or documents using either new or existing models such as the skip-gram model (Mikolov et al., 2013).

Acknowledgments

We would like to thank Abhyuday Jagannatha and the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by the grant HL125089 from the National Institutes of Health (NIH). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

- Samuel R. Bowman, Jon Gauthier, Abhinav Rashtogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *CoRR*, abs/1603.06021.
- Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit³: Web inventory of transcribed and translated talks. In *EAMT*, Trento, Italy.
- Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.
- Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.
- Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *NIPS 2015*, pages 1819–1827.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Ozan Irsoy and Claire Cardie. 2015. Modeling compositionality with multiplicative recurrent neural networks. In *ICLR 2015*.
- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.
- Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR 2014*.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.
- Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML 2014*.
- Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *ICLR 2016*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.
- Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Recognizing entailment and contradiction by tree-based convolution. In *ACL 2016*.
- Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding.
- Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.
- Scott Reed and Nando de Freitas. 2016. Neural programmer-interpreters. In *ICLR 2016*.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR 2016*.
- Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP 2013*.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS 2015*, pages 2431–2439.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.
- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL 2015*.
- Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*.
- Wen tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *ACL 2013*.
- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.
- Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *ICML 2015*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP 2015*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Deep Learning Workshop 2014*.

A Step-by-step visualization of memory states in NSE

Each small table represents the memory state at a single time step. The current time step and input token are listed on the top of the table. The memory slots pointed by the query vector is highlighted in red color. The brackets represent the word composition order in each slot.

<i>t=0</i> <i>input:</i> <S> <S> A little child sits quietly on a hand built rock wall in autumn	<i>t=1</i> <i>input:</i> <S> <S> (<S>A) little child sits quietly on a hand built rock wall in autumn	<i>t=2</i> <i>input:</i> A (A <S>) (<S>A) little child sits quietly on a hand built rock wall in autumn	<i>t=3</i> <i>input:</i> little (A <S>) (<S>A) little child sits (little quietly) on a hand built rock wall in autumn	<i>t=4</i> <i>input:</i> child (A <S>) (<S>A) little child sits (child (little quietly)) on a hand built rock wall in autumn	<i>t=5</i> <i>input:</i> sits (A <S>) (<S>A) little child sits (child (little quietly)) on a hand built rock wall in (sits autumn)
<i>t=6</i> <i>input:</i> quietly (A <S>) (<S>A) little child (quietly sits) (child (little quietly)) on a hand built rock wall in (sits autumn)	<i>t=7</i> <i>input:</i> on (A <S>) (<S>A) little child (quietly sits) (child (little quietly)) on a hand built rock wall in (on (sits autumn))	<i>t=8</i> <i>input:</i> a (A <S>) (<S>A) little child (quietly sits) (child (little quietly)) on a hand (a built) rock wall in (on (sits autumn))	<i>t=9</i> <i>input:</i> hand (A <S>) (<S>A) little child (quietly sits) (child (little quietly)) on a hand (a built) rock (hand wall) in (on (sits autumn))	<i>t=10</i> <i>input:</i> built (A <S>) (<S>A) little child (quietly sits) (child (little quietly)) on a hand (a built) rock (built (hand wall)) in (on (sits autumn))	<i>t=11</i> <i>input:</i> rock (A <S>) (<S>A) little child (quietly sits) (child (little quietly)) on a hand (a built) rock (rock (built (hand wall))) in (on (sits autumn))
<i>t=12</i> <i>input:</i> wall (A <S>) (<S>A) little child (wall (quietly sits)) (child (little quietly)) on a hand (a built) rock (rock (built (hand wall))) in (on (sits autumn))	<i>t=13</i> <i>input:</i> in (A <S>) (<S>A) little child (wall (quietly sits)) (child (little quietly)) on a hand (a built) (in rock) (rock (built (hand wall))) in (on (sits autumn))	<i>t=14</i> <i>input:</i> autumn (A <S>) (<S>A) little child (wall (quietly sits)) (child (little quietly)) on a hand (autumn (a built)) (in rock) (rock (built (hand wall))) in (on (sits autumn))			

Efficient Benchmarking of NLP APIs using Multi-armed Bandits

Gholamreza Haffari and Tuan Tran and Mark Carman

Faculty of Information Technology, Monash University

Clayton, VICTORIA 3800, Australia

{gholamreza.haffari, mark.carman}@monash.edu

tdtra18@student.monash.edu

Abstract

Comparing NLP systems to select the best one for a task of interest, such as named entity recognition, is critical for practitioners and researchers. A rigorous approach involves setting up a hypothesis testing scenario using the performance of the systems on query documents. However, often the hypothesis testing approach needs to send a large number of document queries to the systems, which can be problematic. In this paper, we present an effective alternative based on the multi-armed bandit (MAB). We propose a hierarchical generative model to represent the uncertainty in the performance measures of the competing systems, to be used by Thompson Sampling to solve the resulting MAB. Experimental results on both synthetic and real data show that our approach requires significantly fewer queries compared to the standard benchmarking technique to identify the best system according to F-measure.

1 Introduction

As new NLP systems are (continually) introduced for a task of interest, such as named entity recognition (NER), it is crucial for practitioners and researchers to select the best system. These systems may be designed based on different models and/or learning algorithms. For instance, due to recent advancement in NER research, several NER systems have been proposed and then supported in APIs such as OpenNLP (Ingersoll et al., 2013), Stanford NER (Finkel et al., 2005), ANNIE (Cunningham et al., 2002) and Meaning Cloud (MeaningCloud-LLC, 1998) to name a few.

Often, the competing NLP systems are benchmarked according to their average performance measure, e.g. F-measure capturing both Precision and Recall, across a set of example documents. Each document produces a single F-measure and the true performance of the system is considered to be the expected value across all possible documents from the domain. Performance on individual documents correspond to samples from the performance distribution of the system, and can then be used to determine the best system (or set of systems should the highest performing system not be unique) using rigorous hypothesis testing. However, this approach usually requires querying each competing system with a large number of documents, which can be problematic if either the number of test documents is limited, or the systems are implemented as APIs by a third party and performing each query incurs a cost.

In this paper, we present a statistically effective method to identify the best system from a pool of systems. Our approach requires significantly fewer example documents to reach similar guarantees as the traditional hypothesis testing set up, hence reducing the cost and increasing the speed of inference. Inspired by the previous work (Scott, 2015; Gabillon et al., 2012; Maron and Moore, 1993), We formulate the benchmarking problem as a sequential decision process of choosing the best arm as the results of new queried documents are received. More specifically, our formulation is based on the best arm identification in a multi-armed bandit (MAB) decision process. This allows us to adapt Thompson Sampling (Thompson, 1933) and its variants (Russo, 2016) to efficiently solve the resulting MAB problem.

A crucial difference between the MAB approach and the traditional hypothesis testing is that it is a *sequential* testing process, instead of a static testing process which forces the benchmarker to

wait for a *final answer* at the end of an experiment. As such, we need to model the uncertainty regarding the estimated F-measure of each competing system, and continually update it as each new document is queried. We propose a novel hierarchical model for this purpose, which is generally applicable to document-level evaluation tasks based on F-measure. The inference in our model is done using standard sampling techniques, such as Gibbs sampling.

We analyse empirically the performance of our approach versus the standard hypothesis testing baselines on synthetic datasets as well as real data for the tasks of sentiment classification and named entity recognition. The empirical results confirm that the number of query documents needed to achieve a particular statistical significance level with our approach is much lower than that required by the hypothesis testing baselines.

2 Best Arm Selection in Multi-Armed Bandit

Our aim is to identify the best system among a finite set of systems based on the noisy sequential measurements of their quality. We formulate this problem as the best arm selection in multi-armed bandit. MAB is a sequential decision process where at each time step n an arm a_n from the collection of K slot machines is chosen and played by the gambler. Each arm $a \in \{1, \dots, K\}$ is associated with an *unknown* reward distribution $f(y|\theta_a)$ from which the reward is generated when the arm is pulled. In the best arm selection problem, the gambler’s goal is to select the arm which has the highest expected reward.

In the common formulation of the MAB, the gambler wants to maximise his cumulative rewards. Intuitively, maximising cumulative rewards eventually leads to the selection of the best arm since it is the optimal decision. However, (Bubeck et al., 2009) gives a theoretical analysis that any strategies for optimising cumulative reward is sub-optimal in identifying the best performing arm. To this end, several algorithms have been proposed for the best arm selection e.g. (Maron and Moore, 1993; Gabillon et al., 2012; Russo, 2016). Although originally developed for maximising cumulative rewards, (Chapelle and Li, 2011; Scott, 2015) provide extensive empirical evidence for the practical success of the Thompson Sampling algorithm for the best arm selection. In what follows,

we present Thompson Sampling (TS) and one of its variants, called Pure Exploration TS (PETS), designed specifically for the best arm selection.

2.1 Thompson Sampling

Let us denote by $(a_1, y_1), \dots, (a_T, y_T)$ the sequence of pulled arms and the revealed rewards, a_t is the arm pulled at time step t and y_t is its associated reward. Note that this sequence is continually growing as the experiment progresses and new arms are pulled. Let $f(y|\theta_a)$ be the probability distribution to model the unknown reward function of the arm a . Had we known the parameters of the reward functions, the best arm could then be selected as

$$\operatorname{argmax}_a E_{f(y|\theta_a)}[y] = \operatorname{argmax}_a \int f(y|\theta_a)y dy.$$

Let us denote the collection of all parameters by $\Theta := (\theta_1, \dots, \theta_K)$. Assuming a prior over the parameters $\pi_0(\Theta)$, we take a Bayesian approach and reason about the posterior of the parameters:

$$\pi_T(\Theta) = \frac{\pi_0(\Theta)L_T(\Theta)}{\int_{\Theta'} \pi_0(\Theta')L_T(\Theta')d\Theta'}$$

where Θ is the parameter domain, and $L_T(\Theta)$ is the likelihood of the observed data $\{(a_t, y_t)\}_1^T$

$$L_T(\Theta) := \prod_{t=1}^T f(y_t|\theta_{a_t}).$$

The posterior probability that a particular arm a is optimal (i.e. has the highest expected reward) is:

$$\alpha_{T,a} := \int_{\Theta_a} \pi_T(\Theta)d\Theta$$

where Θ_a is the set of those parameter values under which the arm a would be selected as the optimal arm:

$$\Theta_a := \{\Theta \in \Theta | E_{f(y|\theta_a)} = \operatorname{argmax}_{a'} E_{f(y|\theta_{a'})}\}$$

In Thompson Sampling the next arm to pull is sampled according to the posterior probability of the arm being optimal. That is, an arm a is selected with probability $\alpha_{T,a}$. Efficient implementation of Thompson Sampling generates a sample from $\alpha_{T,a}$ indirectly by first generating a sample from $\pi_T(\Theta)$ and then selecting the next arm to pull by $\operatorname{argmax}_a E_{f(y|\hat{\theta}_a)}[y]$.

Algorithm 1 Pure Exploration TS (PETS)

Initialization: Pull each arm once
 $t = K$
while termination condition is not met **do**
 $\Theta \sim \pi_t(\Theta)$
 $a \leftarrow \operatorname{argmax}_k E_{f(y|\hat{\theta}_k)}[y]$
 $r \sim \operatorname{uniform}(0, 1)$
 if $r \leq \beta$ **then**
 Pull a and update the posterior $\pi_{t+1}(\Theta)$
 else
 $b \leftarrow a$
 while $b = a$ **do**
 $\hat{\Theta} \sim \pi_t(\Theta)$
 $b \leftarrow \operatorname{argmax}_k E_{f(y|\hat{\theta}_k)}[y]$
 end while
 Pull b and update the posterior $\pi_{t+1}(\Theta)$
 end if
 $t \leftarrow t + 1$
end while

2.2 Pure Exploration Thompson Sampling

Thompson sampling can perform poorly for the best arm identification problem. The reason is that once it discovers a particular arm is performing well, it becomes overconfident and almost always selects that arm in the future iterations. In case that arm is not the best arm, it takes a long time for the algorithm to divert from it. For example, if $\alpha_{T,a} = 90\%$, then the algorithm selects an arm other than a on average on every 10 iterations, which would make it significantly longer to get to a point where $\alpha_{T',a'} = 95\%$, i.e. the point where the algorithm terminates with confidence 95% in a different arm a' .

Let $\alpha_T := (\alpha_{T,1}, \dots, \alpha_{T,K})$ be the vector of arm probabilities to be optimal. Pure Exploration Thompson Sampling (Russo, 2016) addresses the above deficiency of Thompson Sampling by throwing away, with probability β , the arm a sampled from α_T . Instead, it samples another arm $b \neq a$ with the probability proportional to $\alpha_{T,b}$. The exploration parameter β prevents the algorithm from exclusively focusing on one arm. Usually β is set to .5 but we empirically investigate other values for this parameter in §4. We can revert to basic Thompson Sampling by setting $\beta = 1$ in PETS. Similar to Thompson Sampling, this arm selection method can be efficiently implemented as shown in Algorithm 1. We terminate the algorithm when it reaches a maximum number of queries or when $\alpha_{T,a} \geq 1 - \delta$, where δ is the confidence parameter provided in the input.

3 A Probabilistic Generative Model of F-measure

In this section we present a novel hierarchical Bayesian model for capturing the uncertainty over systems' F-measures, as the prediction outcome on new query documents are received. We present this model for F-measure, however, we note that it can be extended for other performance measures as well.

F-measure is defined as the harmonic mean of the precision and recall:

$$\begin{aligned} \text{F-measure} &:= \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \\ \text{precision} &:= \frac{TP}{TP + FP}, \quad \text{recall} := \frac{TP}{TP + FN} \end{aligned}$$

where (TP, FP, TN, FN) denote true positive, false positive, true negative, and false negative counts. These counts result from comparing the predictions of a system with the ground truth annotations, and they sum to the total number of annotated data items N . We denote the normalised version of the counts by *rates* $(\hat{TP}, \hat{FP}, \hat{TN}, \hat{FN})$, which are derived by dividing the raw counts by N . Importantly, the rate statistics are enough to calculate precision, recall, and F-measure.

Instead of modelling the uncertainty over the F-measure of a system directly, we model the uncertainty over its rate statistics. Any distribution over $(\hat{TP}, \hat{FP}, \hat{TN}, \hat{FN})$ then induces a distribution over F-measure. The benefit of working with the rate statistics is that they relate more naturally to the observed (TP, FP, TN, FN) counts, as established in our generative model in the following.

More specifically, we assume a hierarchical model to generate the rate statistics of the systems and the observed (TP_d, FP_d, TN_d, FN_d) counts over a collection of documents $d \in \mathcal{D}$. For each system, we draw its rate statistics $\theta_k := (\hat{TP}_k, \hat{FP}_k, \hat{TN}_k, \hat{FN}_k)$ from a Dirichlet prior. To generate the counts statistics resulting from applying the system a_t on the document d_t , we first generate a document-specific rate vector μ_{d_t} from a Dirichlet distribution centred around θ_{a_t} . Note that including explicit document-specific rates μ_{d_t} in the model (from which the binomial counts are drawn) is necessary in order to allow for sufficient variation in the observed error rates across documents, due to the inherent differences in difficulty of labelling different documents.¹

¹In other words, in order to model the observed vari-

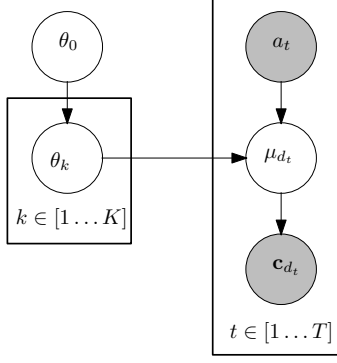


Figure 1: The graphical model for the probabilistic generation of a system’s parameters θ_k and a document’s counts c_{d_t} , as the selected system a_t is applied onto the document d_t at the time step t . The observed quantities are shaded.

We then generate the observed counts $c_{d_t} := (TP_{d_t}, FP_{d_t}, TN_{d_t}, FN_{d_t})$ from the Binomial distribution with parameters μ_{d_t} and N_{d_t} , where N_{d_t} is the number data items in d_t . In summary, the generative model is as follows:

$$\begin{aligned} \forall k \in [1..K] & : \theta_k \sim \text{Dirichlet}(\theta_0, \alpha_0) \\ \forall t \in [1..T] & : \mu_{d_t} \sim \text{Dirichlet}(\theta_{a_t}, \alpha) \\ & c_{d_t} \sim \text{Binomial}(\mu_{d_t}, N_{d_t}) \end{aligned}$$

where α_0 and α are the concentration parameters, which we set to 1 in our experiments in §4. Figure 1 depicts the graphical model.

For inference, the quantities of interest are the unknown rates for the systems $\{\theta_k\}_{k=1}^K$. The observed quantities are document-specific counts $\{c_{d_t}\}_{t=1}^T$, and we would like to marginalise out the latent document-specific rate variables $\{\mu_{d_t}\}_{t=1}^T$. We resort to Gibbs sampling for inference in our model. That is, we iteratively select a hidden variable and sample a value from its posterior given all the other variables are fixed to their current values. In our experiments, we collect 1000 samples from the posterior.² Algorithm 2 depicts the sampling-based inference for the posterior embedded in the PETS algorithm for the best system identification.

F-measure is a frequently used evaluation measure, which can straightforwardly be parametrized to allow for varying the importance of precision

ability in (TP,FP,TN,FN) counts, we had to use a Dirichlet-Compound-Multinomial with shared Dirichlet prior rather than a simple Multinomial with a shared Dirichlet Prior.

²We make use of the JAGS (Just Another Gibbs Sampler) toolkit (Plummer, 2003) for inference in our model.

Algorithm 2 Identifying the best system

Require: K : Number of arms, J : Number of samples from posterior $\pi(\cdot)$, \mathcal{D} : Document collection, $\text{Fmeasure}(\hat{T}P, \hat{T}N, \hat{F}P, \hat{F}N) := \frac{2\hat{T}P}{2\hat{T}P + \hat{F}P + \hat{F}N}$, $\text{NextDoc}(a, \mathcal{D})$: Next document for an arm a from \mathcal{D}

- 1: **for** $k \in [1..K]$ **do**
- 2: $\mathcal{D}_k \leftarrow \text{NextDoc}(k, \mathcal{D})$
- 3: $\mathcal{S}_k \leftarrow \{\tilde{\theta}_j | \forall j \in [1..J] : \tilde{\theta}_j \stackrel{\text{Gibbs}}{\sim} \pi(\theta_j | \mathcal{D}_k)\}$
- 4: **end for**
- 5: **while** termination condition is not met **do**
- 6: **for** $k \in [1..K]$ **do**
- 7: $f_k \sim \{\text{Fmeasure}(\tilde{\theta}) | \tilde{\theta} \in \mathcal{S}_k\}$
- 8: **end for**
- 9: $a \leftarrow \text{argmax}_k f_k$
- 10: $r \sim \text{uniform}(0, 1)$
- 11: $b \leftarrow a$
- 12: **if** $r > \beta$ **then**
- 13: **while** $b = a$ **do**
- 14: **for** $k \in [1..K]$ **do**
- 15: $f_k \sim \{\text{Fmeasure}(\tilde{\theta}) | \tilde{\theta} \in \mathcal{S}_k\}$
- 16: **end for**
- 17: $b \leftarrow \text{argmax}_k f_k$
- 18: **end while**
- 19: **end if**
- 20: $\mathcal{D}_b \leftarrow \mathcal{D}_b \cup \text{NextDoc}(b, \mathcal{D})$
- 21: $\mathcal{S}_b \leftarrow \{\tilde{\theta}_j | \forall j \in [1..J] : \tilde{\theta}_j \stackrel{\text{Gibbs}}{\sim} \pi(\theta_j | \mathcal{D}_b)\}$
- 22: **end while**

versus recall:

$$\text{F}_{\beta}\text{-measure} := \frac{2}{\frac{\beta}{\text{precision}} + \frac{1-\beta}{\text{recall}}}$$

where β is a parameter trading off precision and recall. We note that our approach can be applied straightforwardly to F_{β} -measure to put more weight on precision or recall where appropriate.

4 Empirical Results and Analysis

We designed two sets of experiments to examine the efficiency and performance of each algorithm using synthetic data as well as real data for sentence level sentiment classification and named entity recognition tasks. With the synthetic data, we analyse our probabilistic generative model for F-measure in combination with the arm selection algorithms. With the real data, we showcase the statistical efficiency of our best system identification approach compared to the standard hypothesis testing approach (Demšar, 2006).

In the real data experiments, we define the “best system” as the system with the highest F-measure based on all documents in the collection. The “success rate” in the NER/Sentiment tasks is then simply the percentage of times the best system is correctly identified (i.e. ranked highest when the system selection algorithm is terminated) over multiple runs of the selection algorithm on random

reorderings of the document collection. We emphasise that, in these experiments, we simulate a scenario where the aim is to select the best system with the minimum number of queries to showcase the effectiveness of our approach.

4.1 Baselines

As baselines, we consider the minimum number of documents needed by the standard statistical power approach. The power of a binary hypothesis testing is the probability that the test correctly rejects the null hypothesis (H_0) when the alternative hypothesis (H_1) is true. In order to find a lower-bound for the number of documents, we make use of the power calculation for a paired T-Test.

The T-Test indicates whether or not the difference between two groups’ averages most likely reflects a “real” difference in the population from which the groups were sampled. Assuming we have two competing systems, we can set up a T-Test to assess whether there is a meaningful difference between the F-measures of the two systems.

We assume an efficient experimental design where the same number of (identical) documents are sent to each system. Assuming a typical power setting of 80% and a significance level of 5%, we can calculate an “Oracle baseline” by making use of the true effect size (the standardised difference in mean performance) across the top two systems.³ Obviously this quantity would not be known a-priori of running the experiment, hence the sample size calculated based on this effect size provides a lower-bound on the number of samples that ought to be needed⁴.

Across the systems, average performance on individual documents will vary due to variations in the inherent difficulty of each document. In other words, some documents are harder to label than others. Thus we make use of a paired sample test for the power calculation. Effect sizes are calculated as follows:

- For the synthetic experiments, the variation in difficulty of the documents is not modelled, so we calculate the effect size by simply using the parameters of the simulation as:

³For the power calculation, we use the following R command `power.t.test(delta=effect, sd=1, sig.level=0.05, power=.8, type="paired", alternative="one.sided")`.

⁴Note that the T-Test assumes Gaussian distributed data, but the violation of this assumption is unlikely to greatly affect the sample size estimates.

$\frac{\mu_1 - \mu_2}{\sqrt{(\sigma_1^2 + \sigma_2^2)}}$, where μ_1 is the mean performance on the best system and μ_2 is the mean performance on the second best system (likewise for the standard deviations σ_1 and σ_2).

- For the experiments on real data, variation will be document dependent and hence we calculate the effect size as $\frac{AVG(f_1 - f_2)}{STDEV(f_1 - f_2)}$, where we directly measure the average and standard deviation of the performance differences between the top performing APIs across the documents.

Since we are comparing many APIs at once, and a priori of running the experiment we don’t know which two systems are the best, we make use of two settings for the confidence level (aka P-value threshold) for the power calculation:

- **Baseline₂**: Assume the top two systems are known a priori and use the significance level of 5% directly to produce a lower bound on the number of iterations.
- **Baseline_{K-1}**: Assume one system is new and is being compared with all other $k - 1$ APIs. We reduce the required significance level α using a Bonferroni correction to be $\alpha / (k - 1)$ to take into account the $k - 1$ comparisons being performed.

We stress that this is an unrealistic scenario in which the effect size is known before running the experiment. If this value is not known or needs to be estimated before the experiment a much larger value would be used, for example a value based on the error threshold ϵ might be appropriate.

4.2 Synthetic Data Experiments

Datasets. For the synthetic data, we generate the (TP, FP, TN, FN) counts of applying the competing systems on hypothetical documents, assuming that we know the systems’ *true* rates. An important factor in the difficulty of the problem is the difference in the Fscore of the top two performing systems, which we denote by *margin*. We consider three levels of problem difficulty by considering the margin $m \in \{.01, .025, .05\}$, and for each margin we consider 5 configurations whose competing systems have the specified margin. Having the true $(\hat{TP}, \hat{FP}, \hat{TN}, \hat{FN})$ rates for a competing system, the count statistics for its results on hypothetical documents are

	margin .05		margin .025		margin .01	
	# queries	success%	# queries	success%	# queries	success%
Baseline ₂	120	-	440	-	2725	-
Baseline _{K-1}	185	-	680	-	4195	-
Hierarchical						
+ Thomp. Samp.	22 ± 5	100	41 ± 6	96	66 ± 8	100
+ Pure Eexpl. TS	19 ± 3	100	27 ± 5	100	64 ± 9	96
Gaussian						
+ Thomp. Samp.	513 ± 46	100	1169 ± 84	100	2000 ± 0	100
+ Pure Eexpl. TS	360 ± 33	100	848 ± 63	100	1965 ± 19	100

Table 1: Average number of queries across different margins. The number of systems is 5, and the maximum number of queries is set to 2000, and $\delta = 0.05$.

then generated based on our generative model. For each competing configuration, we repeat the experiment multiple times in order to account for the randomness inherent in the algorithms and the generated documents. In different experiments, we let the number of competing systems K be $\{5, 10, 20\}$.

Margin and the number of systems. In this experiment, we investigate the relation between the margin and the number of documents queried by each algorithm. Intuitively, as the margin between the top performing systems decreases, more queries are required to segregate the best system among the top performing ones. We run each algorithm for 500 times on the competing configurations for each margin with $K = 5$. The maximum number of queries allowed is 2000, and the algorithm can terminate earlier as soon as $\alpha_{T,a} \geq .95$, i.e. $\delta = 0.05$.

Table 1 summarises the average number of queries and the success rates of TS and PETS in combination with our hierarchical Bayesian model for F-measure across different margins. We see that the number of query documents increases as the margin decreases. It is also worth noting that PETS requires slightly smaller number of queries than Thompson Sampling. Interestingly, the number of samples required by the hypothesis testing baselines is much more than that required by the TS/PETS combined with our hierarchical model.

We then ask the question whether the number of competing systems is important. Table 2 summarises the average number of queries and the success rate of each algorithm on the competing configurations for the margin 0.05 for varying number of systems $K \in \{5, 10, 20\}$. As seen, the

number of queries increases (sub)linearly with the number of competing systems.

Hierarchical vs Gaussian. We compare our hierarchical model for capturing the uncertainty over F-measure with the Gaussian distribution. That is, we associate a Gaussian distribution to each system to model its posterior over the F-measure. The use of the Gaussian distribution to model the mean of sampled F-measures is motivated by the law of large numbers. This approach directly models the uncertainty of a system’s F-measure, as opposed to our indirect modelling approach where posterior distribution is constructed using the distribution of $(\hat{F}P, \hat{F}N, \hat{T}P, \hat{T}N)$ rates.

Tables 1 and 2 show the average number of queries and success rates for algorithms using our hierarchical model vs the Gaussian distribution based model. The general trend is that using the Gaussian model in TS/PETS requires significantly more queries compared to the hierarchical model as well as the baselines. Needing more queries compared to the baselines highlights the importance of choosing the right distribution for capturing the uncertainty over the F-measure in TS/PETS. Needing more queries compared to the hierarchical variant is somewhat expected as the synthetic data is generated according to the hierarchical model. However, we will see similar trends in the experiments on the real data.

4.3 Sentiment Analysis

We consider the task of sentence level sentiment prediction for medical documents. The aim is to benchmark systems according to how well they can predict the polarity of sentences contained in a medical report, where each report corresponds

	$K : 5$		$K : 10$		$K : 20$	
	# queries	success%	# queries	success%	# queries	success%
Baseline ₂	120	-	245	-	400	-
Baseline _{$K-1$}	185	-	380	-	620	-
Hierarchical						
+ Thomp. Samp.	22 ± 5	100	43 ± 6	96	70 ± 14	100
+ Pure Eexpl. TS	19 ± 3	100	35 ± 5	100	64 ± 13	100
Gaussian						
+ Thomp. Samp.	513 ± 46	100	1019 ± 59	100	1404 ± 75	100
+ Pure Eexpl. TS	360 ± 33	100	661 ± 38	100	937 ± 62	100

Table 2: Average number of queries across different number of competing systems. The margin is 0.05, and the maximum number of queries is set to 2000, and $\delta = 0.05$.

to a patient.

Dataset. We make use of a biomedical corpus (Martinez et al., 2015) consisting of CT reports for fungal disease detection collected from three hospitals. For each report, only the free text section were used, which contains the radiologist’s understanding of the scan and the reason for the requested scan as written by clinicians. Every report was de-identified: any potentially identifying information such as name, address, age/birthday, gender were removed. There are a total of 358 test documents, where the average number of sentences per document is 23.

Competing Systems. We make use of a variant of the coarse-to-fine model proposed in (McDonald et al., 2007) for sentiment analysis. Briefly speaking, the model couples the sentiment of the sentences contained in a report with the overall sentiment of the report. We train four versions of the model, each of which corresponds to a different training condition:

- M_{full} : where the model is trained on the fully annotated data D_F , i.e. the data annotated at both the sentence and report level.
- M_{partial} : where the model is trained on both D_F and the partially annotated data D_P in which the sentence level annotation is missing but the reports are labeled.
- M_{unlab} : where the model is trained on D_F and D_U in which the annotation is missing at both sentence and report level.
- M_{all} : where the model is trained on all of the available data described above.

	# queries	% success
Baseline ₂	856	-
Baseline _{$K-1$}	1320	-
Hierarchical		
+ Thomp. Samp.	152 ± 36	100
+ Pure Eexpl. TS	123 ± 37	100
Gaussian		
+ Thomp. Samp.	500 ± 0	100
+ Pure Eexpl. TS	500 ± 0	100

Table 3: Sentiment classification for biomedical reports with 4 competing models. The maximum number of queries is set to 500, and $\delta = 0.05$.

We expect M_{all} to outperform the other models. The aim is to analyse the behaviour of our best system selection methods on real data compared to the baselines.

Results. Table 3 presents the results. As seen the number of queries needed for the TS/PETS combined with the hierarchical model is much less than that of the baselines and the Gaussian variant.

4.4 Named Entity Recognition

In our second set of experiment, we attempt to see how our frameworks and F_1 models perform using realistic data.

MASC Corpus. For benchmarking the NER systems, we use the Manually Annotated Sub-Corpus (MASC) (Ide et al., 2008) that includes 19 different domains. The corpus consists of approximately 500K words of contemporary American English written and spoken data drawn from the

Open American National Corpus (OANC). This corpus includes a wide variety of linguistic annotations with a balanced selection of texts from a broad range of genres/domains. The diversity of the corpus will enable us to assess the robustness of tools across different domains. The number of documents in the MASC corpus is about 392.

Competing Systems. We evaluate the performance of 5 popular NER systems implemented as API in third party implementations:

- OpenNLP (Ingersoll et al., 2013): The Apache OpenNLP library is a machine learning based toolkit for the text processing. It is based on the maximum entropy and perceptron.
- Stanford NER (Finkel et al., 2005): It is based on linear chain Conditional Random Field (CRF) sequence models. It is part of the Stanford CoreNLP, which is an integrated suite of NLP tools in Java.
- ANNIE (Cunningham et al., 2002): ANNIE uses gazetteer-based lookups and finite state machines for entity identification and classification. It can recognise persons, locations, organisations, dates, addresses and other named entity types. ANNIE is part of the GATE framework. It can be used as a Web Service but it also provides its own interface for independent use.
- Meaning Cloud (MeaningCloud-LLC, 1998): It is based on a hybrid approach combining machine learning with a rule based system. The software is available as a cloud based solution and on-premise as a plugin module for the GATE framework.
- LingPipe (Alias-i, 2008): It is a set of Java libraries developed by Alias-I for NLP. The NER component is based on a 1st-order Hidden Markov Model with variable-length n -grams as the feature set and uses the IOB annotation scheme for the output.

Results. Table 4 presents the results. As seen the number of queries needed for the TS/PETS combined with the hierarchical model is much less than that of the baselines and the Gaussian variant.

	# queries	% success
Baseline ₂	125	-
Baseline _{K-1}	240	-
Hierarchical		
+ Thomp. Samp.	25 ± 6	99
+ Pure Eexpl. TS	24 ± 5	98
Gaussian		
+ Thomp. Samp.	539 ± 3	100
+ Pure Eexpl. TS	412 ± 2	100

Table 4: Named entity recognition on MASC documents with 5 competing systems. The maximum number of queries is set to 2000, and $\delta = .05$.

5 Conclusion

We have presented a novel approach for benchmarking NLP systems based on the multi-armed bandit (MAB) problem. We have proposed a hierarchical generative model to represent the uncertainty in the performance measures of the competing systems, to be used by the Thompson Sampling algorithm to solve the resulting MAB problem. Experimental results on both synthetic and real data show that our approach requires significantly fewer queries compared to the standard benchmarking technique to identify the best system according to F-measure. Future work includes applying our approach to other NLP problems, particularly emerging document-level problem settings such document-wise machine translation.

Acknowledgments

G. H. appreciates fruitful discussions with Mohammad Ghavamzadeh and Yasin Abbasi-Yadkori. We are grateful to reviewers for their insightful feedback and comments.

References

- Alias-i. 2008. Lingpipe. Available at: <http://alias-i.com/lingpipe>.
- Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. 2009. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, pages 23–37. Springer.
- Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

- Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.
- Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.
- Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. 2012. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220.
- Nancy Ide, Collin Baker, Christiane Fellbaum, and Charles Fillmore. 2008. Masc: The manually annotated sub-corpus of american english. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*. Citeseer.
- Grant S Ingersoll, Thomas S Morton, and Andrew L Farris. 2013. *Taming text: how to find, organize, and manipulate it*. Manning Publications Co.
- Oded Maron and Andrew W Moore. 1993. Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute*, page 263.
- Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 432–439, Prague, Czech Republic, June. Association for Computational Linguistics.
- MeaningCloud-LLC. 1998. Meaning Cloud. Available at: <https://www.meaningcloud.com/>.
- Martyn Plummer. 2003. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*.
- Daniel Russo. 2016. Simple bayesian algorithms for best arm identification. *arXiv preprint arXiv:1602.08448*.
- Steven L Scott. 2015. Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry*, 31(1):37–45.
- W.R. Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3–4):285–294.

Character-Word LSTM Language Models

Lyan Verwimp Joris Pelemans Hugo Van hamme Patrick Wambacq

ESAT – PSI, KU Leuven

Kasteelpark Arenberg 10, 3001 Heverlee, Belgium

firstname.lastname@esat.kuleuven.be

Abstract

We present a Character-Word Long Short-Term Memory Language Model which both reduces the perplexity with respect to a baseline word-level language model and reduces the number of parameters of the model. Character information can reveal structural (dis)similarities between words and can even be used when a word is out-of-vocabulary, thus improving the modeling of infrequent and unknown words. By concatenating word and character embeddings, we achieve up to 2.77% relative improvement on English compared to a baseline model with a similar amount of parameters and 4.57% on Dutch. Moreover, we also outperform baseline word-level models with a larger number of parameters.

1 Introduction

Language models (LMs) play a crucial role in many speech and language processing tasks, among others speech recognition, machine translation and optical character recognition. The current state of the art are recurrent neural network (RNN) based LMs (Mikolov et al., 2010), and more specifically long short-term memory models (LSTM) (Hochreiter and Schmidhuber, 1997) LMs (Sundermeyer et al., 2012) and their variants (e.g. gated recurrent units (GRU) (Cho et al., 2014)). LSTMs and GRUs are usually very similar in performance, with GRU models often even outperforming LSTM models despite the fact that they have less parameters to train. However, Jozefowicz et al. (2015) recently showed that for the task of language modeling LSTMs work better than GRUs, therefore we focus on LSTM-based LMs.

In this work, we address some of the drawbacks of NN based LMs (and many other types of LMs).

A first drawback is the fact that the parameters for infrequent words are typically less accurate because the network requires a lot of training examples to optimize the parameters. The second and most important drawback addressed is the fact that the model does not make use of the internal structure of the words, given that they are encoded as one-hot vectors. For example, ‘felicity’ (great happiness) is a relatively infrequent word (its frequency is much lower compared to the frequency of ‘happiness’ according to Google Ngram Viewer (Michel et al., 2011)) and will probably be an out-of-vocabulary (OOV) word in many applications, but since there are many nouns also ending on ‘ity’ (ability, complexity, creativity . . .), knowledge of the surface form of the word will help in determining that ‘felicity’ is a noun. Hence, subword information can play an important role in improving the representations for infrequent words and even OOV words.

In our character-word (CW) LSTM LM, we concatenate character and word embeddings and feed the resulting character-word embedding to the LSTM. Hence, we provide the LSTM with information about the structure of the word. By concatenating the embeddings, the individual characters (as opposed to e.g. a bag-of-characters approach) are preserved and the order of the characters is implicitly modeled. Moreover, since we keep the total embedding size constant, the ‘word’ embedding shrinks in size and is partly replaced by character embeddings (with a much smaller vocabulary and hence a much smaller embedding matrix), which decreases the number of parameters of the model.

We investigate the influence of the number of characters added, the size of the character embeddings, weight sharing for the characters and the size of the (hidden layer of the) model. Given that common or similar character sequences do not always occur at the beginning of words (e.g. ‘overfitting’ – ‘underfitting’), we also examine adding the charac-

ters in forward order, backward order or both orders.

We test our CW LMs on both English and Dutch. Since Dutch has a richer morphology than English due to among others its productive compounding (see e.g. (Réveil, 2012)), we expect that it should benefit more from a LM augmented with formal/morphological information.

The contributions of this paper are the following:

1. We present a method to combine word and subword information in an LSTM LM: concatenating word and character embeddings. As far as we know, this method has not been investigated before.
2. By decreasing the size of the word-level embedding (and hence the huge word embedding matrix), we effectively reduce the number of parameters in the model (see section 3.3).
3. We find that the CW model both outperforms word-level LMs with the same number of hidden units (and hence a larger number of parameters) and word-level LMs with the same number of parameters. These findings are confirmed for English and Dutch, for a small model size and a large model size. The size of the character embeddings should be proportional to the total size of the embedding (the concatenation of characters should not exceed the size of the word-level embedding), and using characters in the backward order improves the perplexity even more (see sections 3.1, 4.3 and 4.4).
4. The LM improves the modeling of OOV words by exploiting their surface form (see section 4.7).

The remainder of this paper is structured as follows: first, we discuss related work (section 2); then the CW LSTM LM is described (section 3) and tested (section 4). Finally, we give an overview of the results and an outlook to future work (section 5).

2 Related work

Other work that investigates the use of character information in RNN LMs either completely replaces the word-level representation by a character-level one or combines word and character information. Much research has also been done on modeling other types of subword information (e.g. morphemes, syllables), but in this discussion, we limit ourselves to characters as subword information.

Research on replacing the word embeddings entirely has been done for neural machine translation (NMT) by Ling et al. (2015) and Costa-jussà and Fonollosa (2016), who replace word-level embeddings with character-level embeddings. Chung et al. (2016) use a subword-level encoder and a character-level decoder for NMT. In dependency parsing, Ballesteros et al. (2015) achieve improvements by generating character-level embeddings with a bidirectional LSTM. Xie et al. (2016) work on natural language correction and also use an encoder-decoder, but operate for both the encoder and the decoder on the character level.

Character-level word representations can also be generated with convolutional neural networks (CNNs), as Zhang et al. (2015) and Kim et al. (2016) have proven for text classification and language modeling respectively. Kim et al. (2016) achieve state-of-the-art results in language modeling for several languages by combining a character-level CNN with highway (Srivastava et al., 2015) and LSTM layers. However, the major improvement is achieved by adding the highway layers: for a small model size, the purely character-level model without highway layers does not perform better than the word-level model (perplexity of 100.3 compared to 97.6), even though the character model has two hidden layers of 300 LSTM units each and is compared to a word model of two hidden layers of only 200 units (in order to keep the number of parameters similar). For a model of larger size, the character-level LM improves the word baseline (84.6 compared to 85.4), but the largest improvement is achieved by adding two highway layers (78.9). Finally, Jozefowicz et al. (2016) also describe character embeddings generated by a CNN, but they test on the 1B Word Benchmark, a data set of an entirely different scale than the one we use.

Other authors combine the word and character information (as we do in this paper) rather than doing away completely with word inputs. Chen et al. (2015) and Kang et al. (2011) work on models combining words and Chinese characters to learn embeddings. Note however that Chinese characters more closely match subwords or words than phonemes. Bojanowski et al. (2015) operate on the character level but use knowledge about the context words in two variants of character-level RNN LMs. Dos Santos and Zadorzny (2014) join word and character representations in a deep neural network for part-of-speech tagging. Finally, Miyamoto and

Cho (2016) describe a LM that is related to our model, although their character-level embedding is generated by a bidirectional LSTM and we do not use a gate to determine how much of the word and how much of the character embedding is used. However, they only compare to a simple baseline model of 2 LSTM layers of each 200 hidden units without dropout, resulting in a higher baseline perplexity (as mentioned in section 4.3, our CW model also achieves larger improvements than reported in this paper with respect to that baseline).

We can conclude that in various NLP tasks, characters have recently been introduced in several different manners. However, the models investigated in related work are either not tested on a competitive baseline (Miyamoto and Cho, 2016) or do not perform better than our models (Kim et al., 2016). In this paper, we introduce a new and straightforward manner to incorporate characters in a LM that (as far as we know) has not been investigated before.

3 Character-Word LSTM LMs

A word-level LSTM LM works as follows: a word encoded as a one-hot column vector \mathbf{w}_t (at time step t) is fed to the input layer and multiplied with the embedding matrix \mathbf{W}_w , resulting in a word embedding \mathbf{e}_t :

$$\mathbf{e}_t = \mathbf{W}_w \times \mathbf{w}_t \quad (1)$$

The word embedding of the current word \mathbf{e}_t will be the input for a series of non-linear operations in the LSTM layer (we refer to (Zaremba et al., 2015) for more details about the equations of the LSTM cell). In the output layer, probabilities for the next word are calculated based on a softmax function.

In our character-word LSTM LM, the only difference with the baseline LM is the computation of the ‘word’ embedding, which is now the result of word and character input rather than word input only. We concatenate the word embedding with embeddings of the characters occurring in that word:

$$\mathbf{e}_t^\top = [(\mathbf{W}_w \times \mathbf{w}_t)^\top (\mathbf{W}_c^1 \times \mathbf{c}_t^1)^\top \dots (\mathbf{W}_c^n \times \mathbf{c}_t^n)^\top] \quad (2)$$

where \mathbf{c}_t^1 is the one-hot encoding of the first character added, \mathbf{W}_c^1 its embedding matrix and n the total number of characters added to the model. The word \mathbf{w}_t and its characters $\mathbf{c}_t^1, \mathbf{c}_t^2 \dots \mathbf{c}_t^n$ are each projected onto their own embeddings, and the concatenation

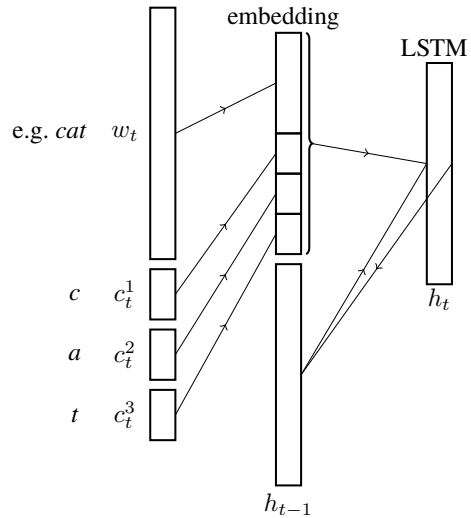


Figure 1: Concatenating word and character embeddings in an LSTM LM.

of the embeddings is the input for the LSTM layer. By concatenating the embeddings, we implicitly preserve the order of the characters: the embedding for e.g. the first character of a word will always correspond to the same portion of the input vector for the LSTM (see figure 1). We also experimented with adding word and character embeddings (a method which does not preserve the order of the characters), but that did not improve the perplexity of the LM.

The number of characters added (n) is fixed. If a word is longer than n characters, only the first (or last, depending on the order in which they are added) n characters are added. If the word is shorter than n , it is padded with a special symbol. Because we can still model the surface form of OOV words with the help of their characters, this model reduces the number of errors made immediately after OOV words (see section 4.7).

3.1 Order of the characters

The characters can be added in the order in which they appear in the word (in the experiments this is called ‘forward order’), in the reversed order (‘backward order’) or both (‘both orders’). In English and Dutch (and many other languages), suffixes can bear meaningful relations (such as plurality and verb conjugation) and compounds typically have word-final heads. Hence, putting more emphasis on the end of a word might help to better model those properties.

3.2 Weight sharing

Note that in equation 2 each position in the word is associated with different weights: the weights for the first character \mathbf{c}_t^1 , \mathbf{W}_c^1 , are different from the weights for the character in the second position, \mathbf{W}_c^2 . Given that the input ‘vocabulary’ for characters is always the same, one could argue that the same set of weights \mathbf{W}_c could be used for all positions in the word:

$$\mathbf{e}_t^\top = [(\mathbf{W}_w \times \mathbf{w}_t)^\top (\mathbf{W}_c \times \mathbf{c}_t^1)^\top \dots (\mathbf{W}_c \times \mathbf{c}_t^n)^\top] \quad (3)$$

However, one could also argue in favor of the opposite case (no shared weights between the characters): for example, an ‘s’ at the end of a word often has a specific meaning, such as indicating a third person singular verb form of the present tense (in English), which it does not have at other positions in the word. Both models with and without weight sharing are tested (see section 4.6).

3.3 Number of parameters

Given that a portion of the total embedding is used for modeling the characters, the actual ‘word’ embedding is smaller which reduces the number of parameters significantly. In a normal word-level LSTM LM, the number of parameters in the embedding matrix is

$$V \times E \quad (4)$$

with V the vocabulary size and $E = E_w$ the total embedding size/word embedding size. In our CW model however, the number of parameters is

$$V \times (E - n \times E_c) + n \times (C \times E_c) \quad (5)$$

with n the number of characters, E_c the size of the character embedding and C the size of the character vocabulary. Since V is by far the dominant factor, reducing the size of the purely word-level embedding vastly reduces the total number of parameters to train. If we share the character weights, that number becomes even smaller:

$$V \times (E - n \times E_c) + C \times E_c \quad (6)$$

4 Experiments

4.1 Setup

All LMs were trained and tested with TensorFlow (Abadi et al., 2015). We test the performance of

the CW architectures for a small model and a large model, with hyperparameters based on Zaremba et al. (2015) and Kim et al. (2016)). The small LSTM consists of 2 layers of 200 hidden units and the large LSTM has 2 layers of 650 hidden units. The total size of the embedding layer always equals the size of the hidden layer. During the first 4/6 (small/large model) epochs, the learning rate is 1, after which we apply an exponential decay:

$$\eta_i = \alpha \eta_{i-1} \quad (7)$$

where η_i is the learning rate at epoch i and α the learning rate decay, which is set to 0.5 for the small LSTM and to 0.8 for the large LSTM. The smaller α , the faster the learning rate decreases. The total number of epochs is fixed to 13/39 (small/large model). During training, 25% of the neurons are dropped (Srivastava et al., 2014) for the small model and 50% for the large model. The weights are randomly initialized to small values (between -0.1 and 0.1 for the small model and between -0.05 and 0.05 for the large model) based on a uniform distribution. We train on mini-batches of 20 with backpropagation through time, where the network is unrolled for 20 steps for the small LSTM and 35 for the large LSTM. The norm of the gradients is clipped at 5 for both models.

For English, we test on the publicly available Penn Treebank (PTB) data set, which contains 900k word tokens for training, 70k word tokens as validation set and 80k words as test set. This data set is small but widely used in related work (among others Zaremba et al. (2015) and Kim et al. (2016)), enabling the comparison between different models. We adopt the same pre-processing as used by previous work (Mikolov et al., 2010) to facilitate comparison, which implies that the dataset contains only lowercase characters (the size of the character vocabulary is 48). Unknown words are mapped to $\langle unk \rangle$, but since we do not have the original text, we cannot use the characters of the unknown words for PTB.

The Dutch data set consists of components g, h, n and o of the Corpus of Spoken Dutch (CGN) (Oostdijk, 2000), containing recordings of meetings, debates, courses, lectures and read text. Approximately 80% was chosen as training set (1.4M word tokens), 10% as validation set (180k word tokens) and 10% as test set (190k word tokens). The size of the Dutch data set is chosen to be similar to the size of the English data set. We also use the same vocabulary size as used for Penn Treebank (10k), since

we want to compare the performance on different languages and exclude any effect of the vocabulary size. However, we do not convert all uppercase characters to lowercase (although the data is normalized such that sentence-initial words with a capital are converted to lowercase if necessary) because the fact that a character is uppercase is meaningful in itself. The character vocabulary size is 88 (Dutch also includes more accented characters due to French loan words, e.g. ‘café’). Hence, we do not only compare two different languages but also models with only lowercase characters and models with both upper- and lowercase characters. Moreover, since we have the original text at our disposal (as opposed to PTB), we can use the characters of the unknown words and still have a character-level representation.

4.2 Baseline models

In our experiments, we compare the CW model with two word-level baselines: one with the same number of hidden units in the LSTM layers (thus containing more parameters) and one with approximately the same number of parameters as the CW model (like Kim et al. (2016) do), because we are interested in both reducing the number of parameters and improving the performance. For the latter baseline, this implies that we change the number of hidden units from 200 to 175 for the small model and from 650 to 475 for the large, keeping the other hyperparameters the same.

The number of parameters for those models is larger than for all CW models except when only 1 or 2 characters are added. The size difference between the CW models and the smaller word-level models becomes larger if more characters are added, if the size of the characters embeddings is larger and if the character weights are shared. The size of the embedding matrix for a word-level LSTM of size 475 is $10,000 \times 475 = 4,750,000$ (V is 10k in all our experiments), whereas for a CW model with 10 character embeddings of size 25 it is of size $10,000 \times (650 - 10 \times 25) + 10 \times (48 \times 25) = 412,000$ (the size of the character vocabulary for PTB is 48), following equation 5. If the character weights are shared, the size of the embedding matrix is only 401,200 (equation 6).

The baseline perplexities for the smaller word-level models are shown in table 1. In the remainder of this paper, ‘ w_x ’ = means word embeddings of size x for a word-level model and ‘ c_x ’ means character embeddings of size x for CW models.

Corpus	Size	Perplexity		
		Validation	Test	
PTB	small	w200	100.7	96.86
		w175	102.62	98.82
	large	w650	87.38	83.6
		w465	88.39	84.38
CGN	small	w200	69.13	76
		w175	69.6	76.78
	large	w650	63.36	70.69
		w475	63.88	70.88

Table 1: Perplexities for the baseline models. Baselines w200 and w650 have the same number of hidden units as the CW models and baselines w175 and w475 approximately have the same number of parameters as the CW models.

4.3 English

In figure 2, the results for a small model trained on Penn Treebank are shown. Almost all CW models outperform the word-based baseline with the same number of parameters (2 LSTM layers of 175 units). Only the CW models in which the concatenated character embeddings take up the majority of the total embedding (more than 7 characters of embedding size 15) perform worse. With respect to the word-level LM with more parameters, only small improvements are obtained. The smaller the character embeddings, the better the performance of the CW model. For example, for a total embedding size of only 200, adding 8 character embeddings of size 15 results in an embedding consisting of 120 units ‘character embedding’ and only 80 units ‘word embedding’, which is not sufficient. The two best performing models add 3 and 7 character embeddings of size 5, giving a perplexity of 100.12 and 100.25 respectively, achieving a relative improvement of 2.44%/2.31% w.r.t. the w175 baseline and 0.58%/0.45% w.r.t. the w200 baseline. For those models, the ‘word embedding’ consists of 185 and 165 units respectively.

We test the performance of the CW architecture on a large model too. In figure 3, the results for different embedding sizes are shown. Just like we saw for the small model, the size of the character embeddings should not be too large: for embeddings of size 50 (‘c50’), the performance drops when a larger number of characters is added. The best result is obtained by adding 8 characters with embeddings of size 25 (‘c25’): a perplexity of 85.97 (2.74%/1.61% relative improvement with respect to

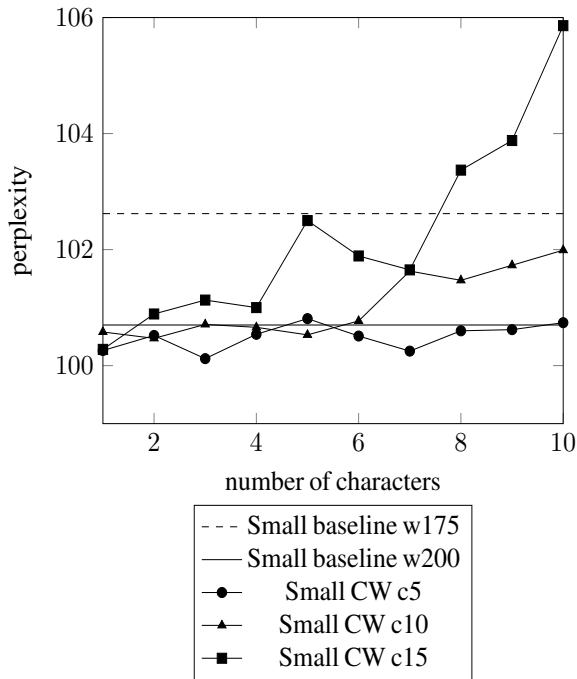


Figure 2: Validation perplexity results on PTB, small model. Different sizes for the character embeddings are tested (‘c5’, ‘c10’, ‘c15’).

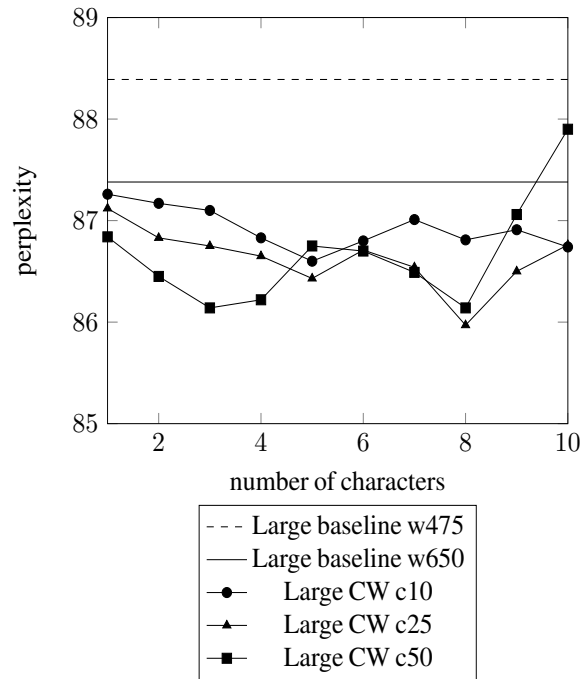


Figure 3: Validation perplexity results on PTB, large model. Different sizes for the character embeddings are tested (‘c10’, ‘c25’, ‘c50’).

the w475/w650 baseline). For embeddings of size 10, adding more than 10 characters gives additional improvements (see figure 4).

We also verify whether the order in which the characters are added is important (figure 4). The best result is achieved by adding the first 3 and the last 3 characters to the model (‘both orders’), giving a perplexity of 85.69, 3.05%/1.87% relative improvement with respect to the w475/w650 baseline. However, adding more characters in both orders causes a decrease in performance. When only adding the characters in the forward order or the backward order, adding the characters in backward order seems to perform slightly better overall (best result: adding 9 characters in the backward order gives a perplexity of 85.7 or 3.04%/1.92% improvement with respect to the w475/w650 baseline).

We can conclude that the size of the character embeddings should be proportional to the total embedding size: the word-level embedding should be larger than the concatenation of the character-level embeddings. Adding characters in the backward order is slightly better than adding them in the forward order, and the largest improvement is made for the large LSTM LM. The test perplexities for some of the best performing models (table 2) confirm these findings.

If we compare the test perplexities with two related models that incorporate characters, we see that our models perform better. Kim et al. (2016) generate character-level embeddings with a convolutional neural network and also report results for both a small and a large model. Their small character-level model has more hidden units than ours (300 compared to 200), but it does not improve with respect to the word-level baseline (since we do not use highway layers, we only compare with the results for models without highway layers). Their large model slightly improves their own baseline perplexity (85.4) by 0.94%. Compare with our results: 2.64% perplexity reduction for the best small LSTM (c5 with $n = 3$) and 2.77% for the best large LSTM (c10 with $n = 3 + 3(b)$). Miyamoto and Cho (2016) only report results for a small model that is trained without dropout, resulting in a baseline perplexity of 115.65. If we train our small model without dropout we get a comparable baseline perplexity (116.33) and a character-word perplexity of 110.54 (compare to 109.05 reported by Miyamoto and Cho (2016)). It remains to be seen whether their model performs equally well compared to better baselines. Moreover, their hybrid character-word model is more complex than ours because it uses a bidirectional LSTM to generate

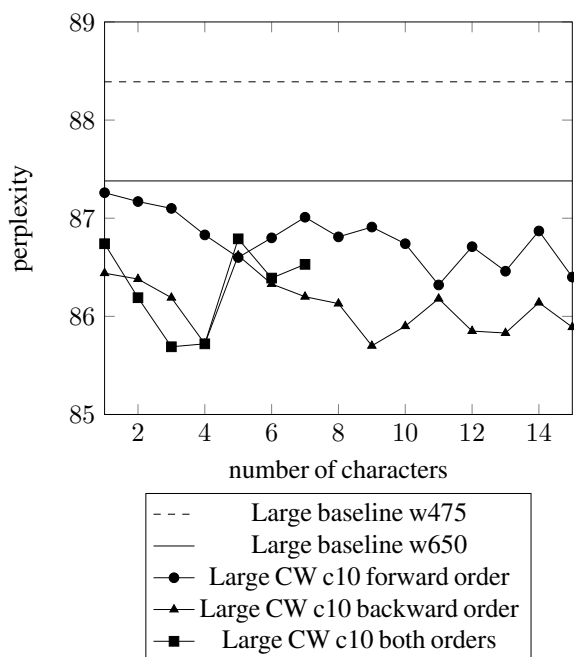


Figure 4: Validation perplexity results on PTB, large model. Several options for the order in which the characters are added are investigated.

the character-level embedding (instead of a lookup table) and a gate to determine the mixing weights between the character- and word-level embeddings.

4.4 Dutch

As we explained in the introduction, we expect that using information about the internal structure of the word will help more for languages with a richer morphology. Although Dutch is still an analytic language (most grammatical relations are marked with separate words or word order rather than morphemes), it has a richer morphology than English because compounding is a productive and widely used process and because it has more lexical variation due to inflection (e.g. verb conjugation, adjective inflection). The results for the LSTM LMs trained on Dutch seem to confirm this hypothesis (see figure 5).

The CW model outperforms the baseline word-level LM both for the small model and the large model. The best result for the small model is obtained by adding 2 or 3 characters, giving a perplexity of 67.59 or 67.65 which equals a relative improvement of 2.89%/2.23% (w175/w200) and 2.80%/2.14% (w175/w200) respectively.

For the large model, we test several embedding sizes and orders for the characters. The best model is the one to which 6 characters in backward order are added, with a perplexity of 60.88 or

Small model	Perplexity
Baseline w175/w200	98.82/96.86
(Kim et al., 2016)	100.3
(Miyamoto and Cho, 2016)	109.05
c5 with $n=3$	96.21
c5 with $n=7$	96.35
Large model	Perplexity
Baseline w475/w650	84.38/83.6
(Kim et al., 2016)	84.6
c25 with $n=8$	82.69
c10 with $n=9(b)$	82.68
c10 with $n=3+3(b)$	82.04

Table 2: Test perplexity results for the best models on PTB. Baseline perplexities are for sizes w175/w200 for a small model and w475/w650 for a large model. n = number of characters added, (b) means backward order. Comparison with other character-level LMs (Kim et al., 2016) (we only compare to models without highway layers) and character-word models (Miyamoto and Cho, 2016) (they do not use dropout and only report results for a small model).

4.70%/3.91% (w475/w650) relative improvement. Just like for PTB, an embedding size of 25 proves to be the best compromise: if the characters are added in the normal order, 4 characters with embeddings of size 25 is the best model (perplexity 61.47 or 3.77%/2.98% (w475/w650) relative improvement).

These results are confirmed for the test set (table 3). The best small model has a perplexity of 75.04 which is 2.27% compared to the baseline and the best large model has a perplexity of 67.64, a relative improvement of 4.57%. The larger improvement for Dutch might be due to the fact that it has a richer morphology and/or the fact that we can use the surface form of the OOV words for the Dutch data set (see sections 4.1 and 4.7).

4.5 Random CW models

In order to investigate whether the improvements of the CW models are not caused by the fact that the characters add some sort of noise to the input, we experiment with adding real noise – random ‘character’ information – rather than the real characters. Both the number of characters (the length of the random ‘word’) and the ‘characters’ themselves are generated based on a uniform distribution. In table 4, the relative change in perplexity, averaged over models to which 1 to 10 characters are added,

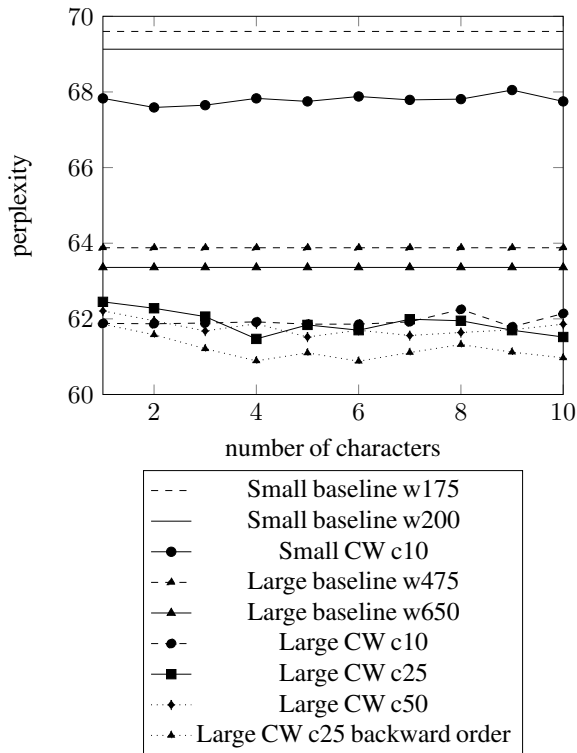


Figure 5: Validation perplexity results on CGN. Several options for the size and order of the character embeddings are investigated.

Small model	Perplexity
Baseline w175/w200	76.78/76
c10 with $n=2$	75.23
c10 with $n=3$	75.04
Large model	Perplexity
Baseline w475/w650	70.88/70.69
c25 with $n=4$	68.79
c25 with $n=6$ (<i>b</i>)	67.64

Table 3: Test perplexity results for the best models on CGN. Baseline perplexities are for sizes w175/w200 for a small model and w475/w650 for a large model. n = number of characters added, (*b*) means backward order.

with respect to the baseline word-level LM and the CW model with real characters is shown.

For English, adding random information had a negative impact on the performance with respect to both the baseline and the CW model. For Dutch on the other hand, adding some random noise to the word-level model gave small improvements. However, the random models perform much worse than the CW models. We can conclude that the characters provide meaningful information to the LM.

		Relative change in valid perplexity w.r.t.	
		Baseline	Char-Word
PTB	small c5	0.34 (0.30)	0.54 (0.46)
	large c15	0.00 (0.29)	0.53 (0.49)
CGN	small c10	-0.18 (0.53)	1.79 (0.47)
	large c10	-0.15 (0.26)	1.52 (1.24)

Table 4: Relative change in validation perplexity for models to which **random** information is added, w.r.t. word-level and CW models. The improvements are averaged over the results for adding 1 to 10 characters/random information, the numbers between brackets are standard deviations. Negative numbers indicate a decrease in perplexity.

4.6 Sharing weights

We repeat certain experiments with the CW models, but with embedding matrices that are shared across all character positions (see section 3.2). Note that sharing the weights does not imply that the position information is lost, because for example the first portion of the character-level embedding always corresponds to the character on the first position. Sharing the weights ensures that a character is always mapped onto the same embedding, regardless of the position of that character in the word, e.g. both occurrences of ‘i’ in ‘felicity’ are represented by the same embedding. This effectively reduces the number of parameters.

We compare the performance of the CW models with weight sharing with the baseline word-level LM and the CW model without weight sharing. In table 5, the relative change with respect to those LMs is listed.

CW models with weight sharing generally improve with respect to a word-level baseline, except for the small English LM. For Dutch, the improvements are more pronounced. The difference with the CW model without weight sharing is small (right column), although *not* sharing the weights works slightly better, which suggests that characters can convey different meanings depending on the position in which they occur. Again, the results are more clear-cut for Dutch than for English.

4.7 Dealing with out-of-vocabulary words

As we mentioned in the introduction, we expect that by providing information about the surface form of OOV words (namely, their characters), the number of errors induced by those words should decrease.

		Relative change in valid perplexity w.r.t.	
		Baseline	Char-Word
PTB	small c10	0.53 (0.88)	0.19 (0.67)
	large c10	- 0.54 (0.37)	- 0.02 (0.22)
CGN	small c10	- 1.70 (0.34)	0.24 (0.30)
	large c10	- 2.10 (0.32)	0.15 (0.50)

Table 5: Relative change in validation perplexity for CW models with **weight sharing** for the characters, w.r.t. baseline and CW models without weight sharing. The improvements are averaged over the results for adding 1 until 10 characters, the numbers between brackets are standard deviations. Negative numbers indicate a decrease in perplexity.

We conduct the following experiment to check whether this is indeed the case: for the CGN test set, we keep track of the probabilities of each word during testing. If an OOV word is encountered, we check the probability of the target word given by a word-level LM and a CW LM. The word-level model is a large model of size 475 and the CW model is a large model in which 6 characters embeddings of size 25 in the backward order are used (the best performing CW model in our experiments).

We observe that in 17,483 of the cases, the CW model assigns a higher probability to the target word following an OOV word, whereas the opposite happens only in 10,724 cases. This is an indication that using the character information indeed helps in better modeling the OOV words.

5 Conclusion and future work

We investigated a character-word LSTM language model, which combines character and word information by concatenating the respective embeddings. This both reduces the size of the LSTM and improves the perplexity with respect to a baseline word-level LM. The model was tested on English and Dutch, for different model sizes, several embedding sizes for the characters, different orders in which the characters are added and for weight sharing of the characters. We can conclude that for almost all setups, the CW LM outperforms the word-level model, whereas it has fewer parameters than the word-level model with the same number of LSTM units. If we compare with a word-level LM with approximately the same number of parameters, the improvement is larger.

One might argue that using a CNN or an RNN

to generate character-level embeddings is a more general approach to incorporate characters in a LM, but this model is simple, easier to train and smaller. Moreover, related models using a CNN-based character embedding (Kim et al., 2016) do not perform better.

For both English and Dutch, we see that the size of the character embedding is important and should be proportional to the total embedding size: the total size of the concatenated character embeddings should not be larger than the word-level embedding. Not using the characters in the order in which they appear in the word, but in the reversed order (and hence putting more emphasis on the end of the word), performs slightly better, although adding only a few characters both from the beginning and the end of the word achieves good performance too.

Using random inputs instead of the characters performed worse than using the characters themselves, thus refuting the hypothesis that the characters simply introduce noise. Sharing the weights/embedding matrices for the characters reduces the size of the model even more, but causes a small increase in perplexity with respect to a model without weight sharing. Finally, we observe that the CW models are better able to deal with OOV words than word-level LMs.

In future work, we will test other architectures to incorporate character information in a word-level LSTM LM, such as combining a character-level LSTM with a word-level LSTM. Another representation that might be useful uses character co-occurrence vectors (by analogy with the acoustic co-occurrences used by Van hamme (2008; 2012)) rather than one-hot character vectors, because co-occurrences intrinsically give information about the order of the characters. Other models could be more inspired by human language processing: according to the theory of *blocking*, we humans have both a mental lexicon of frequent words and a morphological module that is used to process infrequent/ unknown words or to create new words (see e.g. (Aronoff and Anshen, 2001)). This could correspond to a word-level LM for frequent words and a subword-level LM for infrequent words.

Acknowledgments

This work was funded by IWT-INNOVATIEF AANBESTEDEN and VRT in the STON project.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Geoffrey Irving, Andrew Harp, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow.org*.
- Marc Aronoff and Frank Anshen. 2001. Morphology and the lexicon: Lexicalization and productivity. In Andrew Spencer and Arnold M. Zwicky, editors, *The Handbook of Morphology*, pages 237–247. Blackwell Publishing.
- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-Based Parsing by Modeling characters instead of words with LSTMs. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 349–359.
- Piotr Bojanowski, Armand Joulin, and Tomáš Mikolov. 2015. Alternative structures for character-level RNNs. *arXiv:1511.06303*.
- Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint Learning of Character and Word Embeddings. *Conference on Artificial Intelligence (AAI)*, pages 1236–1242.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation. *arXiv:1603.06147*.
- Marta R. Costa-jussà and José A.R. Fonollosa. 2016. Character-based Neural Machine Translation. *Proceedings of the Association for Computational Linguistics (ACL)*, 2:357–361.
- Cícero N. dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-Speech Tagging. *Proceedings of The 31st International Conference on Machine Learning (ICML)*, pages 1818–1826.
- Hugo Van hamme. 2008. HAC-models: a novel approach to continuous speech recognition. *Proceedings Interspeech*, pages 2554–2557.
- Hugo Van hamme. 2012. An on-line NMF model for temporal pattern learning: Theory with application to automatic speech recognition. *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 306–313.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350.
- Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *arXiv:1602.02410*.
- Moonyoung Kang, Tim Ng, and Long Nguyen. 2011. Mandarin word-character hybrid-input Neural Network Language Model. *Proceedings Interspeech*, pages 1261–1264.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. *Proc. Conference on Artificial Intelligence (AAAI)*, pages 2741–2749.
- Wang Ling, Isabel Trancoso, Chris Dyer, and Alan Black. 2015. Character-based Neural Machine Translation. *arXiv:1511.04586*.
- Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, William Brockman, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, 331(6014):176–182.
- Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *Proceedings Interspeech*, pages 1045–1048.
- Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated Word-Character Recurrent Language Model. *arXiv:1606.01700*.
- Nelleke Oostdijk. 2000. The Spoken Dutch Corpus. Overview and first Evaluation. *Proceedings Language Resources and Evaluation Conference (LREC)*, pages 887–894.
- Bert Réveil. 2012. Optimizing the Recognition Lexicon for Automatic Speech Recognition. *PhD thesis, University of Ghent, Belgium*.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

- Rupesh K. Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training Very Deep Networks. *Neural Information Processing Systems Conference (NIPS)*, pages 2377–2385.
- Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM Neural Networks for Language Modeling. *Proceedings Interspeech*, pages 1724–1734.
- Ziang Xie, Anand Avati, Naveen Arivzhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural Language Correction with Character-Based Attention. *arXiv:1603.09727*.
- Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent Neural Network Regularization. *arXiv:1409.2329*.
- Xiang Zhang, Junbo Zhao, and Yann LeCunn. 2015. Character-level Convolutional Networks for Text Classification. *Neural Information Processing Systems Conference (NIPS)*, pages 649–657.

A Hierarchical Neural Model for Learning Sequences of Dialogue Acts

Quan Hung Tran and Ingrid Zukerman and Gholamreza Haffari

Faculty of Information Technology, Monash University

Clayton, VICTORIA 3800, Australia

{hung.tran, ingrid.zukerman, gholamreza.haffari}@monash.edu

Abstract

We propose a novel hierarchical Recurrent Neural Network (RNN) for learning sequences of Dialogue Acts (DAs). The input in this task is a sequence of utterances (i.e., conversational contributions) comprising a sequence of tokens, and the output is a sequence of DA labels (one label per utterance). Our model leverages the hierarchical nature of dialogue data by using two nested RNNs that capture long-range dependencies at the dialogue level and the utterance level. This model is combined with an attention mechanism that focuses on salient tokens in utterances. Our experimental results show that our model outperforms strong baselines on two popular datasets, Switchboard and MapTask; and our detailed empirical analysis highlights the impact of each aspect of our model.

1 Introduction

The sequence-labeling task involves learning a model that maps an input sequence to an output sequence. Many NLP problems can be treated as sequence-labeling tasks, e.g., part-of-speech (PoS) tagging (Toutanova et al., 2003; Toutanova and Manning, 2000), machine translation (Brown et al., 1993) and automatic speech recognition (Gales and Young, 2008). Recurrent Neural Nets (RNNs) have been the workhorse model for many NLP sequence-labeling tasks, e.g., machine translation (Sutskever et al., 2014) and speech recognition (Amodei et al., 2015), due to their ability to capture long-range dependencies inherent in natural language.

In this paper, we propose a hierarchical RNN for labeling a sequence of utterances (i.e., contributions) in a dialogue with their Dialogue Acts

(DAs). This task is particularly useful for dialogue systems, as knowing the DA of an utterance supports its interpretation, and the generation of an appropriate response. The DA classification problem differs from the aforementioned tasks in the structure of the input and the immediacy of the output. The input in these tasks is a sequence of tokens, e.g., a sequence of words in PoS tagging; while in DA classification, the input is hierarchical, i.e., a conversation comprises a sequence of utterances, each of which has a sequence of tokens (Figure 1). In addition, to be useful for dialogue systems, the DA of an utterance must be determined immediately, hence a bi-directional approach is not feasible.

As mentioned above, RNNs are able to capture long-range dependencies. This ability was harnessed by Shen and Lee (2016) for DA classification. However, they ignored the conversational dimension of the data, treating the utterances in a dialogue as separate instances — an assumption that results in loss of information. To overcome this limitation, we designed a two-layer RNN model that leverages the hierarchical nature of dialogue data: an outer-layer RNN encodes the conversational dimension, and an inner-layer RNN encodes the utterance dimension.

One of the difficulties of sequence labeling is that different elements of an input sequence have different degrees of importance for the task at hand (Shen and Lee, 2016), and the noise introduced by less important elements might degrade the performance of a labeling model. To address this problem, we incorporate into our model the attention mechanism described in (Shen and Lee, 2016), which has yielded performance improvements in DA classification compared to traditional RNNs.

Our empirical results show that our *hierarchical RNN model with an attentional mechanism* out-

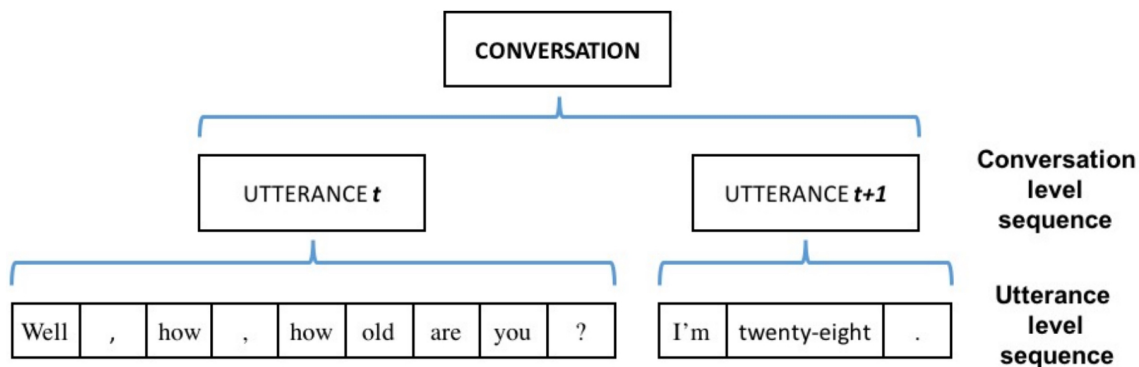


Figure 1: Switchboard data example.

performs strong baselines on two popular datasets: Switchboard (Jurafsky et al., 1997; Stolcke et al., 2000) and MapTask (Anderson et al., 1991). In addition, we provide an empirical analysis of the impact of the main aspects of our model on performance: utterance RNN, conversation RNN, and information source for the attention mechanism.

This paper is organised as follows. In the next section, we discuss related research in DA classification. In Section 3, we describe our RNN. Our experiments and results are presented in Section 4, followed by our analysis and concluding remarks.

2 Related Research

Independent DA classification. In this approach, each utterance is treated as a separate instance, which allows the application of general classification algorithms. Julia *et al.* (2010) employed a Support Vector Machine (SVM) with n-gram features obtained from an utterance-level Hidden Markov Model (HMM) to ascribe DAs to audio signals and textual transcriptions of the MapTask corpus. Webb *et al.* (2005) used a similar approach, employing cue phrases as features.

Sequence-based DA classification. This approach takes advantage of the sequential nature of conversations. In one of the earliest works in DA classification, Stolcke *et al.* (2000) used an HMM with a trigram language model to classify DAs in the Switchboard corpus, achieving an accuracy of 71.0%. In this work, the trigram language model was employed to calculate the symbol emission probability of the HMM. Surendran *et al.* (2006) also used an HMM, but employed output symbol probabilities produced by an SVM classifier, instead of emission probabilities obtained from

a trigram language model. More recently, the Recurrent Convolutional Neural Network model proposed by Kalchbrenner and Blunsom (2013) achieved an accuracy of 73.9% on the Switchboard corpus. In this work, a Convolutional Neural Network encodes each utterance into a vector, which is then treated as input to a conversation-level RNN. The DA is then classified using a softmax layer applied on top of the hidden states of the RNN.

Attention in Neural Models. Attentional Neural Models have been successfully applied to sequence-to-sequence mapping tasks, notably machine translation and DA classification. Bahdanau *et al.* (2014) proposed an attentional encoder-decoder architecture for machine translation. The encoder encodes the input sequence into a sequence of hidden vectors; the decoder decodes the information stored in the hidden sequence to generate the output; and the attentional mechanism is used to summarize a sentence into a context vector dynamically, helping the decoder decide which part of the sequence to attend to when generating a target word. As mentioned above, Shen and Lee (2016) employed an attentional RNN for independent DA classification; they achieved an accuracy of 72.6% on textual transcriptions of the Switchboard corpus.

3 Model Description

Suppose we have a sequence of observations $\mathbf{o} := \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_m\}$ and the corresponding sequence of labels $\mathbf{y} := \{y_1, y_2, \dots, y_m\}$, where each observation \mathbf{o}_t is a sequence. Our hierarchical-attentional model, denoted *HA-RNN*, learns the conditional probability $P(\mathbf{y}|\mathbf{o})$ relating the ob-

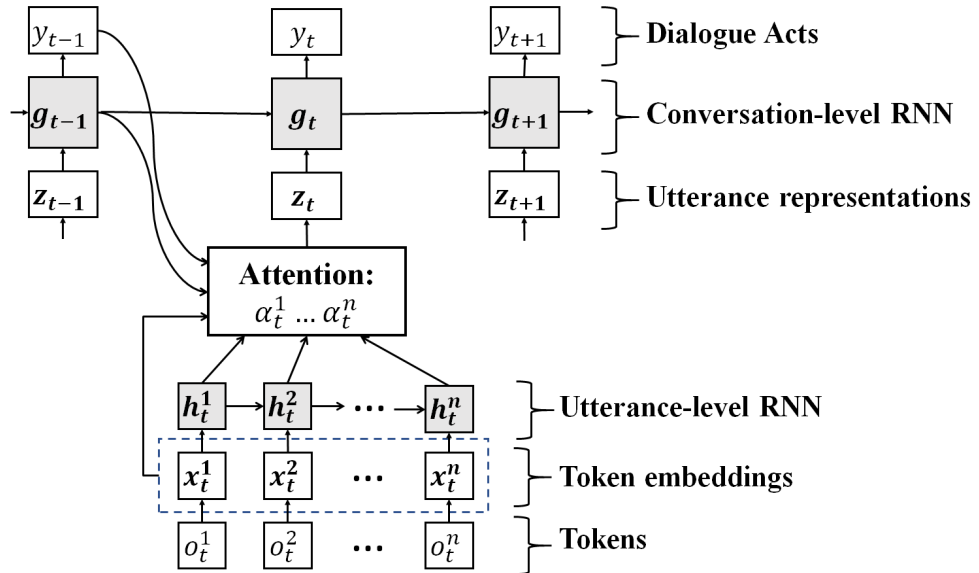


Figure 2: HA-RNN – Hierarchical-attentional RNN model.

served sequence to its label sequence, based on the following decomposition:

$$\begin{aligned}
 P(\{y_1, y_2, \dots, y_m\} | \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_m\}) \\
 = \prod_{t=1}^m P(y_t | \mathbf{y}_{<t}, \mathbf{o}_{\leq t}) \quad (1)
 \end{aligned}$$

Note that our model conditions on the full history, rather than a finite history as done in Markov models, such as maximum entropy Markov models (McCallum et al., 2000).

We employ neural networks to model the constituent conditional distributions. Our model comprises three main elements (Figure 2): (1) an *utterance-level RNN* that encodes the information within the utterances; (2) an *attentional mechanism* that highlights the important parts of an input utterance, and summarizes the information within the utterance into a real-valued vector; and (3) a *conversation-level RNN* that encodes the information of the whole dialogue sequence. As discussed in Section 1, our hierarchical-RNN design was motivated by the structure of the input data, while the attentional mechanism has proven to be effective in DA classification (Shen and Lee, 2016).

Utterance-level RNN. This RNN was implemented using LSTM (Hochreiter and Schmidhuber, 1997; Graves, 2013). First, an embedding matrix maps each token (e.g., word or punctuation marker) into a dense vector representation. Let us denote the sequence of tokens in the t -th utterance as $\mathbf{o}_t := \{o_t^1, o_t^2, \dots, o_t^n\}$, which is

mapped into the sequence of embedding vectors $\mathbf{x}_t := \{x_t^1, x_t^2, \dots, x_t^n\}$ using the token embedding table \mathbf{w} :

$$\mathbf{x}_t^i = \mathbf{e}_w(o_t^i) \quad (2)$$

The utterance RNN then takes as input this sequence of vectors, and produces a sequence of corresponding hidden vectors $\mathbf{h}_t = \{h_t^1, h_t^2, \dots, h_t^n\}$, which capture the information within the tokens, and put the tokens in their sentential context:

$$\mathbf{h}_t^i = \text{RNN}_{utter}(\mathbf{h}_t^{i-1}, \mathbf{x}_t^i) \quad (3)$$

The parameters of the utterance RNN and the token embeddings are learned during training.

Attentional mechanism. This mechanism summarizes the hidden vectors of the utterance-level RNN into a single vector representing the whole utterance. The attention vector is a sequence of positive numbers that sum to 1, where each number corresponds to a token in an utterance, and represents the importance of the token for understanding the DA associated with the utterance. The final representation \mathbf{z}_t of the t -th utterance is the sum of the corresponding elements of its hidden vectors weighted by attention weights:

$$\mathbf{z}_t = \sum_i \alpha_t^i \mathbf{h}_t^i \quad (4)$$

We posit that the main factors for determining the importance of a token for DA classification are: (1) the meaning of the token, as represented

by its embedding vector; and (2) the full context of the conversation, particularly the previous DA. For example, if the DA of an utterance is *Yes-No-Question*, and there is a “yes” or “no” token in the next utterance, this token is likely to be important. Equation 5 integrates these factors to compute attention scores:

$$s_t^i = \mathbf{U} \cdot \tanh(W^{(\text{in})} \cdot \mathbf{x}_t^i + W^{(\text{co})} \cdot \mathbf{g}_{t-1} + \mathbf{e}_a(y_{t-1}) + \mathbf{b}^{(\text{in})}) \quad (5)$$

where vector $\mathbf{e}_a(y_{t-1})$ denotes the embedding of the previous DA, which is similar to the embedding of tokens; and vector \mathbf{g}_{t-1} is the previous hidden vector of the conversation-level RNN, detailed below, which summarizes the conversation so far. $W^{(\text{in})}$ and $W^{(\text{co})}$ are parameter matrices for the input tokens and the conversational context respectively, and \mathbf{U} and $\mathbf{b}^{(\text{in})}$ are parameter vectors — all of which are learned during training. The scores s_t^i are mapped into a probability vector by means of a softmax function:

$$\boldsymbol{\alpha}_t = \text{softmax}(\mathbf{s}_t) \quad (6)$$

Conversation-level RNN. This RNN is structurally similar to the utterance-level RNN. The input to the conversation-level RNN is the sequence of vectors \mathbf{z} generated for the utterances in a conversation, which is then encoded by the RNN into a sequence of hidden vectors \mathbf{g} :

$$\mathbf{g}_t = \text{RNN}_{\text{convers}}(\mathbf{g}_{t-1}, \mathbf{z}_t) \quad (7)$$

This information is then used in the generation of the output DA:

$$y_t | \mathbf{y}_{<t}, \mathbf{o}_{\leq t} \sim \text{softmax}(\mathbf{W}^{(\text{out})} \cdot \mathbf{g}_t + \mathbf{b}^{(\text{out})}) \quad (8)$$

where the matrix $\mathbf{W}^{(\text{out})}$, vector $\mathbf{b}^{(\text{out})}$ and the parameters of the conversation-level network $\text{RNN}_{\text{convers}}$ are learned during the training.

During testing, ideally a given sequence of observed utterances \mathbf{o} should be decoded to a label sequence \mathbf{y} that maximizes the conditional probability $P(\mathbf{y}|\mathbf{o})$ according to the model. However, finding the highest-scoring label sequence is a computationally hard problem, since the conversation-level RNN does not lend itself to dynamic programming. Therefore, we employ a greedy decoding approach, where, going left-to-right, at each step we choose the y_t with the highest probability in the local DA distribution. This

method is common practice in sequence-labeling RNNs, e.g., in neural machine translation (Bahdanau et al., 2014; Sutskever et al., 2014; Luong et al., 2015).

4 Experiments

4.1 Data sets

We tested our models on the Switchboard corpus (Jurafsky et al., 1997; Stolcke et al., 2000) and the MapTask corpus (Anderson et al., 1991) — two popular datasets used for DA classification. At this stage of our research, we consider only transcriptions of the conversations in both corpora (the incorporation of phonetic input (Taylor et al., 1998; Wright Hastie et al., 2002; Julia et al., 2010) is the subject of future work). Thus, we compare our results only with those obtained by systems that employ transcriptions exclusively.

Switchboard corpus. This corpus contains DA-annotated transcriptions of 1155 telephone conversations with no specific topic, which have an average of 176 utterances. Originally, there were approximately 226 DA tags in the corpus, but in the DA classification literature, the tags are usually clustered into 42 tags.¹ Table 1(a) shows percentages of the seven most frequent tags in the data. Following (Stolcke et al., 2000), in our experiments we use 1115 conversations for training, 21 for development and 19 for testing.

MapTask corpus. This is a richly annotated corpus that comprises 128 dialogues about instruction following, containing 212 utterances on average. Each conversation has an instruction giver and an instruction follower. The instruction giver gives directions with reference to a map, which the instruction follower must follow. The MapTask corpus has 13 DA tags, including the “unclassifiable” tag. Table 1(b) shows percentages of the seven most frequent tags in the data. We randomly split this data into 80% training, 10% development and 10% test sets, which contain 103, 12 and 13 conversations respectively.

4.2 Results

We experimented with different embedding sizes and hidden layer dimensions for our model *HA-RNN*, and selected the following, which yielded

¹The official manual stated that there were originally 220 tags. We follow the tag-clustering procedure by Christopher Potts described in comprag.christopherpotts.net/swda.html.

DA tag	Example	Percentage	DA tag	Example	Percentage
<i>Statement-non-opinion</i>	I'm twenty-eight	36%	<i>Acknowledge</i>	Mmhmm	21%
<i>Acknowledge (Backchannel)</i>	Uh-huh	19%	<i>Instruct</i>	And we're finished	16%
<i>Statement-opinion</i>	I think it's great	13%	<i>Reply_y</i>	Yeah	12%
<i>Agree/Accept</i>	That's exactly it	5%	<i>Explain</i>	I've got a bridge	8%
<i>Abandoned or Turn-Exit</i>	So,	5%	<i>Check</i>	Is that it	8%
<i>Appreciation</i>	I can imagine.	2%	<i>Ready</i>	And then	8%
<i>Yes-No-Question</i>	Do you?	2%	<i>Align</i>	See what i mean	7%

(a) Switchboard

(b) MapTask

Table 1: Seven most frequent DAs and examples for (a) Switchboard and (b) MapTask.

Model	Accuracy
<i>RCNN</i>	73.9%
<i>RNN-Attentional-C</i>	72.6%
<i>HMM-trigram-C</i>	71.0%
<i>HA-RNN</i>	74.5%

Table 2: Performance on Switchboard.

Model	Accuracy
<i>HMM-trigram-C</i>	52.3%
<i>Random Forest</i>	52.5%
<i>Random Forest + prev DA</i>	55.3%
<i>HA-RNN</i>	63.3%

Table 3: Performance on MapTask.

the best performance with reasonable run times. The word-embedding size was set to 250, and the DA-embedding size to 180. The hidden dimension of the utterance-level RNN was set to 160, and the hidden dimension of the conversation-level RNN was set to 250. Our model was implemented with the CNN package.² During training, the negative log-likelihood was optimized using Adagrad (Duchi et al., 2011), with dropout rate 0.5 to prevent over-fitting (Srivastava et al., 2014). Training terminated when the log-likelihood of the development set did not improve. As mentioned in Section 3, during testing, the sequence of output labels was generated with greedy decoding. Statistical significance was computed on the MapTask test data using McNemar’s test with $\alpha = 0.05$ (we could not compute statistical significance for the Switchboard results, because they were obtained from the literature, and we did not have access to per-conversation labels).

Switchboard. We compare our model’s performance with that of the following strong baselines: (*RCNN*) the recurrent convolutional neural network model from (Kalchbrenner and Blunsom, 2013); (*RNN-Attentional-C*) the attention-based RNN classifier from (Shen and Lee, 2016); and (*HMM-trigram-C*) the HMM-based classifier from (Stolcke et al., 2000).

The results in Table 2 show that our model outperforms these baselines.³ The higher ac-

²github.com/clab/cnn.

³Two other works on Switchboard DA classification (Gambäck et al., 2011; Webb and Ferguson, 2010) used experimental setups that differ from ours, respectively obtaining

curacy of our model compared to classifier-based approaches (i.e., *RNN-Attentional-C* and *HMM-trigram-C*) confirms that taking into account dependencies among the DAs through the conversation-level RNN improves accuracy. Furthermore, the better performance of our model compared to *RCNN* shows that summarizing utterances with an RNN augmented with an attention architecture is more effective than using a convolution architecture for DA sequence labeling.

MapTask. Due to the unavailability of standard training/development/test sets for this dataset, we compare the results obtained by our model with those obtained by our implementation of the following independent DA classifiers: *HMM-trigram-C* (Stolcke et al., 2000); *Random Forest* – an instance-based random forest classifier; and *Random Forest + prev DA* – a random forest classifier that uses the previous DA tag.

The results in Table 3 show that our model outperforms these baselines (statistically significant). These results reinforce the insights from the Switchboard corpus, whereby taking into account conversational dependencies between DAs substantially improves DA-labeling performance.⁴

accuracies of 77.85% and 80.72%. However, these results are not directly comparable to Stolcke *et al.*’s (2000) or ours, and are therefore excluded from our comparison.

⁴Two studies on MapTask DA classification were performed under experimental setups that differ from ours: Julia *et al.* (2010) employed *HMM+SVM* on text transcriptions and audio signals, obtaining an accuracy of 55.4% for transcriptions only. Surendran and Levow (2006) used *Viterbi+SVM*, posting a classification accuracy of 59.1% for transcriptions — the best result among systems that employ transcription data exclusively. Unfortunately, Julia *et al.*’s de-

5 Analysis

5.1 Architectural analysis

We investigate the influence of the main components of our model on performance by creating variants of our model through the addition or removal of connections or layers. We then compare the performance of these variants with that of the original model in terms of DA-classification accuracy and negative log-likelihood on the test, development and training partitions of our datasets. As done in Section 4, statistical significance is calculated for the test partitions of both datasets using McNemar’s test with $\alpha = 0.05$.

Does an RNN at the utterance level help? To answer this question, we create a variant, denoted *woUttRNN*, where attentional coefficients are applied directly to the token embeddings. Thus, Equation 4 is changed to Equation 9:

$$\mathbf{z}_t = \sum_i \alpha_t^i \mathbf{x}_t^i \quad (9)$$

As seen in Tables 4 and 5, removing the utterance-level RNN (*woUttRNN*) reduces the accuracy and increases the negative log likelihood for the training, development and test partitions of both datasets. These changes are statistically significant for the test set.

Which sources of information are critical for computing the attentional component? In our main model, *HA-RNN*, we calculate the attentional signal using information from the previous DA, the previous hidden vector representation of the conversation-level RNN, and the embeddings of the tokens. To determine the contribution of the first two resources to the performance of the model, we create two variants of *HA-RNN*: *woDA2Attn*, which employs only the previous conversation-level RNN hidden vector; and *woHid2Attn*, which employs only the previous DA. Thus, in *woDA2Attn*, Equation 5 becomes Equation 10, and in *woHid2Attn*, Equation 5 becomes Equation 11:

$$s_t^i = \mathbf{U} \cdot \tanh(W^{(\text{in})} \cdot \mathbf{x}_t^i + W^{(\text{co})} \cdot \mathbf{g}_{t-1} + \mathbf{b}^{(\text{in})}) \quad (10)$$

$$s_t^i = \mathbf{U} \cdot \tanh(W^{(\text{in})} \cdot \mathbf{x}_t^i + \mathbf{e}_a(y_{t-1}) + \mathbf{b}^{(\text{in})}) \quad (11)$$

scription of their MapTask subset is not sufficient to replicate their experiment, and Surendran and Levow’s data split is not accessible. Notwithstanding the difference in conditions, our model’s accuracy is superior to theirs.

As seen in Tables 4 and 5, both of these resources provide valuable information, but the changes in performance due to the omission of these resources are smaller than those obtained with *woUttRNN*. Removing the DA connection (*woDA2Attn*) or the previous conversation-level RNN hidden vector (*woHid2Attn*) leads to statistically significant drops in accuracy and increases in negative log-likelihood on the test partitions of both datasets. The changes in performance with respect to the development and training sets vary across the datasets. As seen in Table 4, both models exhibit accuracy drops (and small increases in negative log-likelihood) on the Switchboard development set, but small accuracy increases (and negative log-likelihood drops) on the Switchboard training set — an indication of over-fitting. In contrast, as seen in Table 5, both models yield a negligible or no drop in accuracy on the MapTask development set, while both yield a drop in accuracy on the training set.

How important is the RNN at the conversation level? To answer this question, we create a variant of our *HA-RNN* model, denoted *woConvRNN*, where the recurrent connections between the units in the conversation-level RNN are removed. The LSTM basis function is calculated with a fixed vector \mathbf{g}_0 instead of the previous time step’s vector. Thus Equation 7 becomes Equation 12:

$$\mathbf{g}_t = \mathbf{f}(\mathbf{g}_0, \mathbf{z}_t) \quad (12)$$

As seen in Tables 4 and 5, *HA-RNN* outperforms *woConvRNN* on the training/development/test partitions of both datasets. The difference between the performance of *HA-RNN* and *woConvRNN* is statistically significant for the test set.

How effective are the DA connections? We have seen that the DA connections improve our model’s performance when they are used to calculate the attentional signal. However, intuitively, the previous DA can also directly provide information about the current DA. For example, it is often the case that a *Yes-No-Question* is followed by *Reply.y* or *Reply.n*. To reflect this observation, we create another model, denoted *wDA2DA*, that has an additional direct connection between the previous DA and the current DA. That is, Equation 8 becomes Equation 13:

$$y_t | \mathbf{y}_{<t}, \mathbf{o}_{\leq t} \sim \text{softmax}(\mathbf{W}^{(\text{out})} \cdot \mathbf{g}_t + \mathbf{e}_o(y_{t-1}) + \mathbf{b}^{(\text{out})}) \quad (13)$$

	Accuracy			Neg log likelihood		
	Test	Dev	Train	Test	Dev	Train
<i>HA-RNN</i>	74.5%	76.2%	80.0%	3770	2819	130333
<i>woUttRNN</i>	71.8%	73.3%	77.4%	4542	3474	163350
<i>woDA2Attn</i>	72.7%	74.3%	80.5%	3835	2932	127445
<i>woHid2Attn</i>	72.8%	75.0%	81.0%	4024	2917	124483
<i>woConvRNN</i>	71.8%	74.1%	76.7%	4537	3648	165734
<i>wDA2DA</i>	71.0%	72.7%	79.2%	4737	3884	150436

Table 4: Performance of variants of the *HA-RNN* model on Switchboard.

	Accuracy			Neg log likelihood		
	Test	Dev	Train	Test	Dev	Train
<i>HA-RNN</i>	63.3%	61.9%	73.4%	3486	3228	18191
<i>woUttRNN</i>	56.9%	58.0%	62.2%	3823	3445	25074
<i>woDA2Attn</i>	61.4%	61.7%	70.1%	3539	3212	19780
<i>woHid2Attn</i>	62.2%	61.9%	71.8%	3487	3248	19132
<i>woConvRNN</i>	58.9%	60.0%	66.9%	3579	3248	20961
<i>wDA2DA</i>	58.2%	58.4%	69.3%	4014	3663	21135

Table 5: Performance of variants of the *HA-RNN* model on MapTask.

As seen in Tables 4 and 5, *wDA2DA* performs much worse than *HA-RNN*. We posit that this happens due to the *exposure bias* problem (Ranzato et al., 2015). That is, during training, the model has access to the correct DA of the previous utterance. However, during testing, the decoding process has access only to predicted DAs, which may lead to the propagation of errors. To quantify the effect of this problem on our model, we designed another experiment where the variants of our model can access the correct DA even during testing; the results for the test partitions of both datasets appear in Table 6.

The results in Table 6 show that exposure bias has different effects on the different variants of our model. As expected, *woDA2Attn*, which does not consider the previous DA, exhibits no change in performance between the oracle and greedy conditions. The models that employ a DA connection to compute the attention signal (*HA-RNN*, *woUttRNN*, *woHid2Attn*, *woConvRNN*) show a slight improvement in accuracy when using the correct DA as input, instead of the predicted DA. In contrast, *wDA2DA* shows large improvements when using the correct DA (3.5% on Switchboard and 6.8% on MapTask), becoming the best-performing model for both datasets. This improvement may

be attributed to the direct connection between the DAs in this model, which increases the influence of previous DAs on the prediction of the current DA — previous DA predictions that are largely correct will substantially improve the performance of *wDA2DA*, while noisy DA predictions will have the opposite effect.

5.2 Attentional Analysis

We analyze how our model *HA-RNN* distributes attention over the tokens in an utterance in order to identify tokens in focus.

Figure 3 shows how the attentional vector highlights the most important tokens in sample utterances in the context of the DA-classification task. For example, in “yes I do”, the most important token that identifies the *Reply_y* class is the token “yes”, which receives most of the probability mass from the attention mechanism.

Table 7 shows the most attended tokens for four classes of DA in MapTask. We compiled these lists by computing the average attention that a token received for all the utterances in a DA class (we excluded tokens that appear less than 5 times). As shown in Table 7, both important tokens “move” and “yes” in Figure 3 appear in their respective DA columns. Two of the most common

	Switchboard		MapTask	
	Oracle	Greedy	Oracle	Greedy
HA-RNN	74.6%	74.5%	64.1%	63.3%
<i>woUttRNN</i>	73.2%	71.8%	56.9%	57.1%
<i>woDA2Attn</i>	73.7%	73.7%	61.4%	61.4%
<i>woHid2Attn</i>	73.8%	72.8%	62.4%	62.2%
<i>woConvRNN</i>	72.2%	71.8%	58.9%	58.9%
<i>wDA2DA</i>	75.0%	71.5%	65.0%	58.2%

Table 6: Performance of oracle and greedy decoding on Switchboard and MapTask test data.

instruct	<s>	move	right	across	the	page	</s>
explain	<s>	i	haven't	got	that	</s>	
align	<s>	un--	go	underneath	it	yeah	</s>
query_w	<s>	where's	the	machete	</s>		
reply_w	<s>	that's	in	the	middle	of	the two </s>
reply_no	<s>	not	in	that	corner	</s>	
query_yes/no	<s>	have	you	got	anything	down	that side </s>
reply_yes	<s>	yes	i	do	</s>		
clarify	<s>	you're	staying	well	below	that	</s>
acknowledge	<s>	meadow	yeah	uh-huh	</s>		
check	<s>	is	that	right	over	in	the right-hand side </s>
explain	<s>	that	means	i've	passed	the	bar </s>

Figure 3: Sample DAs with highlighted attention vectors for MapTask.

<i>Acknowledge</i>	<i>Instruct</i>	<i>Reply_y</i>	<i>Reply_n</i>
mmhmm	move	mmhmm	nope
uh-huh	continue	uh-huh	i've
yes	drop	yes	no
yeah	starting	yep	it's
see	pass	aye	you
go	reach	i've	go
aye	stop	yeah	don't
no	coming	i'm	not
you	go	you	haven't
i'm	whatever	go	just

Table 7: Sample DA-specific high-focus tokens for MapTask.

labels, *Acknowledge* and *Reply_y*, have very similar attended tokens. In fact, many utterances in *Acknowledge* and *Reply_y* have the same text form. Thus, the distinction between the two classes is

highly dependent upon the conversational context. Also, note that although *Reply_n* is not one of the most common DAs in MapTask, our model can still learn the most important tokens for this DA.

6 Conclusions

In this paper, we proposed a novel hierarchical RNN for learning sequences of DAs. Our model leverages the hierarchical nature of dialogue data by using two nested RNNs that capture long-range dependencies at the conversation level and the utterance level. We further combine the model with an attention mechanism to focus on salient tokens in utterances. Our experimental results show that our model outperforms strong baselines on two popular datasets: Switchboard and MapTask. In the future, we plan to address the exposure bias problem, and incorporate acoustic features and speaker information into our model.

Acknowledgments

This research was supported in part by grant DP120100103 from the Australian Research Council.

References

- Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. 2015. Deep speech 2: End-to-end speech recognition in English and Mandarin. *arXiv preprint arXiv:1512.02595*.
- Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Steven Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC MapTask corpus. *Language and speech*, 34(4):351–366.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.
- Mark Gales and Steve Young. 2008. The application of hidden Markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304.
- Björn Gambäck, Fredrik Olsson, and Oscar Täckström. 2011. Active learning for dialogue act classification. In *Proceedings of Interspeech 2011*, pages 1329–1332, Florence, Italy.
- Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Fatema N. Julia, Khan M. Iftekharuddin, and Atiq U. Islam. 2010. Dialog act classification using acoustic and discourse information of MapTask data. *International Journal of Computational Intelligence and Applications*, 9(4):289–311.
- Daniel Jurafsky, Elizabeth Shriberg, and Debra Bisca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. Technical report, Stanford University.
- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.
- Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP’2015 – Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.
- Andrew McCallum, Dayne Freitag, and Fernando C.N. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *ICML’00 – Proceedings of the 17th International Conference on Machine Learning*, pages 591–598, Stanford, California.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.
- Sheng-syun Shen and Hung-yi Lee. 2016. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. *arXiv preprint arXiv:1604.00077*.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.
- Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.
- Dinoj Surendran and Gina-Anne Levow. 2006. Dialog act tagging with Support Vector Machines and hidden Markov models. In *Proceedings of Interspeech 2006*, pages 1950–1953, Pittsburgh, Pennsylvania.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Paul A. Taylor, Simon King, Steve D. Isard, and Helen Wright Hastie. 1998. Intonation and dialogue context as constraints for speech recognition. *Language and Speech*, 41(3-4):493–512.
- Kristina Toutanova and Christopher D Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 63–70, Hong Kong.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network.

In *NAACL'2003 – Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Edmonton, Canada.

Nick Webb and Michael Ferguson. 2010. Automatic extraction of cue phrases for cross-corpus dialogue act classification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1310–1317, Uppsala, Sweden.

Nick Webb, Mark Hepple, and Yorick Wilks. 2005. Dialogue act classification based on intra-utterance features. In *Proceedings of the AAAI Workshop on Spoken Language Understanding*, Pittsburgh, Pennsylvania.

Helen Wright Hastie, Massimo Poesio, and Steve D. Isard. 2002. Automatically predicting dialogue structure using prosodic features. *Speech Communication*, 36(1):63–79.

A Network-based End-to-End Trainable Task-oriented Dialogue System

Tsung-Hsien Wen¹, David Vandyke¹, Nikola Mrkšić¹, Milica Gašić¹,
Lina M. Rojas-Barahona¹, Pei-Hao Su¹, Stefan Ultes¹, and Steve Young¹

¹Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK

{thw28, djv27, nm480, mg436, lmr46, phs26, su259, sjy11}@cam.ac.uk

Abstract

Teaching machines to accomplish tasks by conversing naturally with humans is challenging. Currently, developing task-oriented dialogue systems requires creating multiple components and typically this involves either a large amount of handcrafting, or acquiring costly labelled datasets to solve a statistical learning problem for each component. In this work we introduce a neural network-based text-in, text-out end-to-end trainable goal-oriented dialogue system along with a new way of collecting dialogue data based on a novel pipe-lined Wizard-of-Oz framework. This approach allows us to develop dialogue systems easily and without making too many assumptions about the task at hand. The results show that the model can converse with human subjects naturally whilst helping them to accomplish tasks in a restaurant search domain.

1 Introduction

Building a task-oriented dialogue system such as a hotel booking or a technical support service is difficult because it is application-specific and there is usually limited availability of training data. To mitigate this problem, recent machine learning approaches to task-oriented dialogue system design have cast the problem as a partially observable Markov Decision Process (POMDP) (Young et al., 2013) with the aim of using reinforcement learning (RL) to train dialogue policies online through interactions with real users (Gašić et al., 2013). However, the language understanding (Henderson et al., 2014; Yao et al., 2014) and language generation (Wen et al., 2015b; Wen et al., 2016) modules still rely on supervised learning and therefore

need corpora to train on. Furthermore, to make RL tractable, the state and action space must be carefully designed (Young et al., 2013; Young et al., 2010), which may restrict the expressive power and learnability of the model. Also, the reward functions needed to train such models are difficult to design and hard to measure at run-time (Su et al., 2015; Su et al., 2016).

At the other end of the spectrum, sequence to sequence learning (Sutskever et al., 2014) has inspired several efforts to build end-to-end trainable, non-task-oriented conversational systems (Vinyals and Le, 2015; Shang et al., 2015; Serban et al., 2015b). This family of approaches treats dialogue as a source to target sequence transduction problem, applying an encoder network (Cho et al., 2014) to encode a user query into a distributed vector representing its semantics, which then conditions a decoder network to generate each system response. These models typically require a large amount of data to train. They allow the creation of effective chatbot type systems but they lack any capability for supporting domain specific tasks, for example, being able to interact with databases (Sukhbaatar et al., 2015; Yin et al., 2015) and aggregate useful information into their responses.

In this work, we propose a neural network-based model for task-oriented dialogue systems by balancing the strengths and the weaknesses of the two research communities: the model is end-to-end trainable¹ but still modularly connected; it does not directly model the user goal, but nevertheless, it still learns to accomplish the required task by providing *relevant* and *appropriate* responses at each turn; it has an explicit representation of database (DB) attributes (slot-value pairs) which it uses to achieve a high task success rate, but has a distributed representation of user intent (dialogue act)

¹We define *end-to-end trainable* as that each system module is trainable from data except for a database operator.

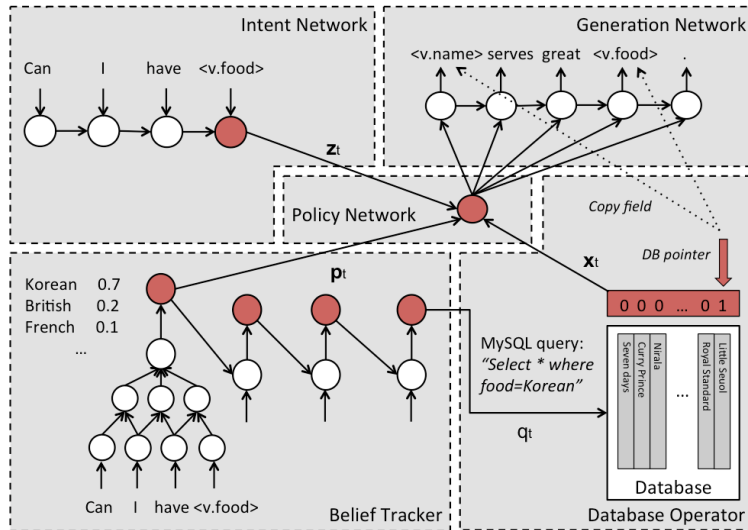


Figure 1: The proposed end-to-end trainable dialogue system framework

to allow ambiguous inputs; and it uses dellexicalisation² and a weight tying strategy (Henderson et al., 2014) to reduce the data required to train the model, but still maintains a high degree of freedom should larger amounts of data become available. We show that the proposed model performs a given task very competitively across several metrics when trained on only a few hundred dialogues.

In order to train the model for the target application, we introduce a novel pipe-lined data collection mechanism inspired by the Wizard-of-Oz paradigm (Kelley, 1984) to collect human-human dialogue corpora via crowd-sourcing. We found that this process is simple and enables fast data collection online with very low development costs.

2 Model

We treat dialogue as a sequence to sequence mapping problem (modelled by a sequence-to-sequence architecture (Sutskever et al., 2014)) augmented with the dialogue history (modelled by a set of belief trackers (Henderson et al., 2014)) and the current database search outcome (modelled by a database operator), as shown in Figure 1. At each turn, the system takes a sequence of tokens² from the user as input and converts it into two internal representations: a distributed representation generated by an intent network and a probability distribution over slot-value pairs called the belief state (Young et al., 2013) generated by a set of belief trackers. The database operator then selects the

²Dellexicalisation: we replaced slots and values by generic tokens (e.g. keywords like Chinese or Indian are replaced by <v.food> in Figure 1) to allow weight sharing.

most probable values in the belief state to form a query to the DB, and the search result, along with the intent representation and belief state are transformed and combined by a policy network to form a single vector representing the next system action. This system action vector is then used to condition a response generation network (Wen et al., 2015a; Wen et al., 2015b) which generates the required system output token by token in skeletal form. The final system response is then formed by substituting the actual values of the database entries into the skeletal sentence structure. A more detailed description of each component is given below.

2.1 Intent Network

The intent network can be viewed as the encoder in the sequence-to-sequence learning framework (Sutskever et al., 2014) whose job is to encode a sequence of input tokens $w_0^t, w_1^t, \dots, w_N^t$ into a distributed vector representation \mathbf{z}_t at every turn t . Typically, a Long Short-term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is used and the last time step hidden layer \mathbf{z}_t^N is taken as the representation,

$$\mathbf{z}_t = \mathbf{z}_t^N = \text{LSTM}(w_0^t, w_1^t, \dots, w_N^t) \quad (1)$$

Alternatively, a convolutional neural network (CNN) can be used in place of the LSTM as the encoder (Kalchbrenner et al., 2014; Kim, 2014),

$$\mathbf{z}_t = \text{CNN}(w_0^t, w_1^t, \dots, w_N^t) \quad (2)$$

and here we investigate both. Since all the slot-value specific information is dellexicalised, the encoded vector can be viewed as a distributed intent

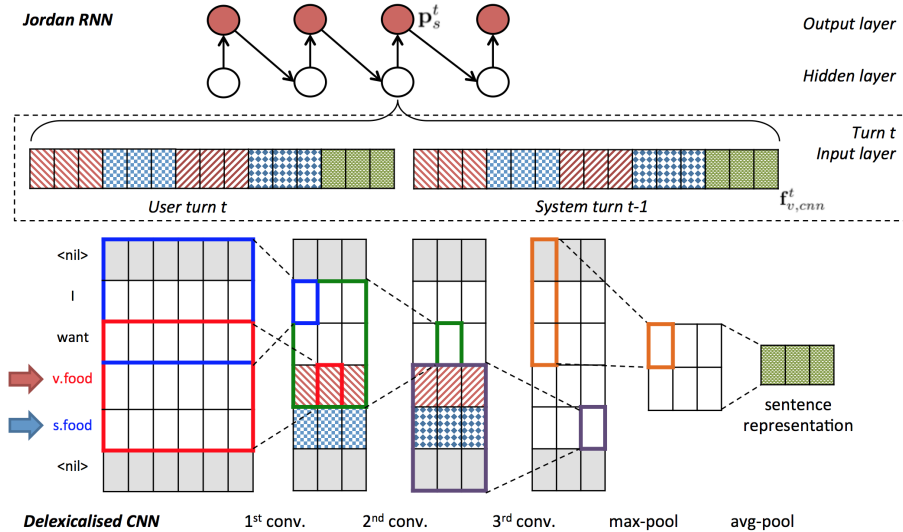


Figure 2: Tied Jordan-type RNN belief tracker with delexicalised CNN feature extractor. The output of the CNN feature extractor is a concatenation of top-level sentence (green) embedding and several levels of intermediate ngram-like embeddings (red and blue). However, if a value cannot be delexicalised in the input, its ngram-like embeddings will all be padded with zeros. We pad zero vectors (in gray) before each convolution operation to make sure the representation at each layer has the same length. The output of each tracker \mathbf{p}_s^t is a distribution over values of a particular slot s .

representation which replaces the hand-coded dialogue act representation (Traum, 1999) in traditional task-oriented dialogue systems.

2.2 Belief Trackers

Belief tracking (also called Dialogue State tracking) provides the core of a task-oriented spoken dialogue system (SDS) (Henderson, 2015). Current state-of-the-art belief trackers use discriminative models such as recurrent neural networks (RNN) (Mikolov et al., 2010; Wen et al., 2013) to directly map ASR hypotheses to belief states (Henderson et al., 2014; Mrkšić et al., 2016). Although in this work we focus on text-based dialogue systems, we retain belief tracking at the core of our system because: (1) it enables a sequence of free-form natural language sentences to be mapped into a fixed set of slot-value pairs, which can then be used to query a DB. This can be viewed as a simple version of a semantic parser (Berant et al., 2013); (2) by keeping track of the dialogue state, it avoids learning unnecessarily complicated long-term dependencies from raw inputs; (3) it uses a smart weight tying strategy that can greatly reduce the data required to train the model, and (4) it provides an inherent robustness which simplifies future extension to spoken systems.

Using each user input as new evidence, the task of a belief tracker is to maintain a multinomial dis-

tribution p over values $v \in V_s$ for each informable slot s , and a binary distribution for each requestable slot³. Each slot in the ontology \mathbb{G}^4 has its own specialised tracker, and each tracker is a Jordan-type (recurrence from output to hidden layer) (Jordan, 1989) RNN⁵ with a CNN feature extractor, as shown in Figure 2. Like Mrkšić et al. (2015), we tie the RNN weights together for each value v but vary features \mathbf{f}_v^t when updating each pre-softmax activation g_v^t . The update equations for a given slot s are,

$$\mathbf{f}_v^t = \mathbf{f}_{v,cnn}^t \oplus p_v^{t-1} \oplus p_{\emptyset}^{t-1} \quad (3)$$

$$g_v^t = \mathbf{w}_s \cdot \text{sigmoid}(\mathbf{W}_s \mathbf{f}_v^t + \mathbf{b}_s) + b'_s \quad (4)$$

$$p_v^t = \frac{\exp(g_v^t)}{\exp(g_{\emptyset,s}) + \sum_{v' \in V_s} \exp(g_{v'}^t)} \quad (5)$$

where vector \mathbf{w}_s , matrix \mathbf{W}_s , bias terms \mathbf{b}_s and b'_s , and scalar $g_{\emptyset,s}$ are parameters. p_{\emptyset}^t is the probability that the user has not mentioned that slot up to turn t and can be calculated by substituting $g_{\emptyset,s}$ for g_v^t in the numerator of Equation 5. In order to model the discourse context at each turn, the feature vector

³Informable slots are slots that users can use to constrain the search, such as food type or price range; Requestable slots are slots that users can ask a value for, such as address.

⁴A small knowledge graph defining the slot-value pairs the system can talk about for a particular task.

⁵We don't use the recurrent connection for requestable slots since they don't need to be tracked.

$\mathbf{f}_{v,cnn}^t$ is the concatenation of two CNN derived features, one from processing the user input u_t at turn t and the other from processing the machine response m_{t-1} at turn $t - 1$,

$$\mathbf{f}_{v,cnn}^t = \text{CNN}_{s,v}^{(u)}(u_t) \oplus \text{CNN}_{s,v}^{(m)}(m_{t-1}) \quad (6)$$

where every token in u_t and m_{t-1} is represented by an embedding of size N derived from a 1-hot input vector. In order to make the tracker aware when delexicalisation is applied to a slot or value, the slot-value specialised CNN operator $\text{CNN}_{s,v}^{(\cdot)}(\cdot)$ extracts not only the top level sentence representation but also intermediate n-gram-like embeddings determined by the position of the delexicalised token in each utterance. If multiple matches are observed, the corresponding embeddings are summed. On the other hand, if there is no match for a particular slot or value, the empty n-gram embeddings are padded with zeros. In order to keep track of the position of delexicalised tokens, both sides of the sentence are padded with zeros before each convolution operation. The number of vectors is determined by the filter size at each layer. The overall process of extracting several layers of position-specific features is visualised in Figure 2.

The belief tracker described above is based on Henderson et al. (2014) with some modifications: (1) only probabilities over informable and requestable slots and values are output, (2) the recurrent memory block is removed, since it appears to offer no benefit in this task, and (3) the n-gram feature extractor is replaced by the CNN extractor described above. By introducing slot-based belief trackers, we essentially add a set of intermediate labels into the system as compared to training a pure end-to-end system. Later in the paper we will show that these tracker components are critical for achieving task success. We will also show that the additional annotation requirement that they introduce can be successfully mitigated using a novel pipe-lined Wizard-of-Oz data collection framework.

2.3 Policy Network and Database Operator

Database Operator Based on the output \mathbf{p}_s^t of the belief trackers, the DB query q_t is formed by,

$$q_t = \bigcup_{s' \in S_I} \{\text{argmax}_v \mathbf{p}_{s'}^t\} \quad (7)$$

where S_I is the set of informable slots. This query is then applied to the DB to create a binary truth

value vector \mathbf{x}_t over DB entities where a 1 indicates that the corresponding entity is consistent with the query (and hence it is consistent with the most likely belief state). In addition, if \mathbf{x} is not entirely null, an associated entity pointer is maintained which identifies one of the matching entities selected at random. The entity pointer is updated if the current entity no longer matches the search criteria; otherwise it stays the same. The entity referenced by the entity pointer is used to form the final system response as described in Section 2.4.

Policy network The policy network can be viewed as the glue which binds the system modules together. Its output is a single vector \mathbf{o}_t representing the system action, and its inputs are comprised of \mathbf{z}_t from the intent network, the belief state \mathbf{p}_s^t , and the DB truth value vector \mathbf{x}_t . Since the generation network only generates appropriate sentence forms, the individual probabilities of the categorical values in the informable belief state are immaterial and are summed together to form a summary belief vector for each slot $\hat{\mathbf{p}}_s^t$ represented by three components: the summed value probabilities, the probability that the user said they "don't care" about this slot and the probability that the slot has not been mentioned. Similarly for the truth value vector \mathbf{x}_t , the number of matching entities matters but not their identity. This vector is therefore compressed to a 6-bin 1-hot encoding $\hat{\mathbf{x}}_t$, which represents different degrees of matching in the DB (no match, 1 match, ... or more than 5 matches). Finally, the policy network output is generated by a three-way matrix transformation,

$$\mathbf{o}_t = \tanh(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{po}\hat{\mathbf{p}}_t + \mathbf{W}_{xo}\hat{\mathbf{x}}_t) \quad (8)$$

where matrices \mathbf{W}_{zo} , \mathbf{W}_{po} , and \mathbf{W}_{xo} are parameters and $\hat{\mathbf{p}}_t = \bigoplus_{s \in \mathbb{G}} \hat{\mathbf{p}}_s^t$ is a concatenation of all summary belief vectors.

2.4 Generation Network

The generation network uses the action vector \mathbf{o}_t to condition a language generator (Wen et al., 2015b). This generates template-like sentences token by token based on the language model probabilities,

$$P(w_{j+1}^t | w_j^t, \mathbf{h}_{j-1}^t, \mathbf{o}_t) = \text{LSTM}_j(w_j^t, \mathbf{h}_{j-1}^t, \mathbf{o}_t) \quad (9)$$

where $\text{LSTM}_j(\cdot)$ is a conditional LSTM operator for one output step j , w_j^t is the last output token (i.e. a word, a delexicalised slot name or a delexicalised

slot value), and \mathbf{h}_{j-1}^t is the hidden layer. Once the output token sequence has been generated, the generic tokens are replaced by their actual values: (1) replacing delexicalised slots by random sampling from a list of surface forms, e.g. $\langle s.food \rangle$ to *food* or *type of food*, and (2) replacing delexicalised values by the actual attribute values of the entity currently selected by the DB pointer. This is similar in spirit to the Latent Predictor Network (Ling et al., 2016) where the token generation process is augmented by a set of pointer networks (Vinyals et al., 2015) to transfer entity specific information into the response.

Attentive Generation Network Instead of decoding responses directly from a static action vector \mathbf{o}_t , an attention-based mechanism (Bahdanau et al., 2014; Hermann et al., 2015) can be used to dynamically aggregate source embeddings at each output step j . In this work we explore the use of an attention mechanism to combine the tracker belief states, i.e. \mathbf{o}_t is computed at each output step j by,

$$\mathbf{o}_t^{(j)} = \tanh(\mathbf{W}_{zo}\mathbf{z}_t + \hat{\mathbf{p}}_t^{(j)} + \mathbf{W}_{xo}\hat{\mathbf{x}}_t) \quad (10)$$

where for a given ontology \mathbb{G} ,

$$\hat{\mathbf{p}}_t^{(j)} = \sum_{s \in \mathbb{G}} \alpha_s^{(j)} \tanh(\mathbf{W}_{po}^s \cdot \hat{\mathbf{p}}_s^t) \quad (11)$$

and where the attention weights $\alpha_s^{(j)}$ are calculated by a scoring function,

$$\alpha_s^{(j)} = \text{softmax}(\mathbf{r}^\top \tanh(\mathbf{W}_r \cdot \mathbf{u}_t)) \quad (12)$$

where $\mathbf{u}_t = \mathbf{z}_t \oplus \hat{\mathbf{x}}_t \oplus \hat{\mathbf{p}}_s^t \oplus \mathbf{w}_j^t \oplus \mathbf{h}_{j-1}^t$, matrix \mathbf{W}_r , and vector \mathbf{r} are parameters to learn and \mathbf{w}_j^t is the embedding of token w_j^t .

3 Wizard-of-Oz Data Collection

Arguably the greatest bottleneck for statistical approaches to dialogue system development is the collection of appropriate training data, and this is especially true for task-oriented dialogue systems. Serban et al (Serban et al., 2015a) have catalogued existing corpora for developing conversational agents. Such corpora may be useful for bootstrapping, but, for task-oriented dialogue systems, in-domain data is essential⁶. To mitigate this problem, we propose a novel crowdsourcing version of the Wizard-of-Oz (WOZ) paradigm (Kelley, 1984) for collecting domain-specific corpora.

⁶E.g. technical support for Apple computers may differ completely from that for Windows, due to the many differences in software and hardware.

Based on the given ontology, we designed two webpages on Amazon Mechanical Turk, one for wizards and the other for users (see Figure 4 and 5 for the designs). The users are given a task specifying the characteristics of a particular entity that they must find (e.g. *a Chinese restaurant in the north*) and asked to type in natural language sentences to fulfil the task. The wizards are given a form to record the information conveyed in the last user turn (e.g. *pricerange=Chinese, area=north*) and a search table showing all the available matching entities in the database. Note these forms contain all the labels needed to train the slot-based belief trackers. The table is automatically updated every time the wizard submits new information. Based on the updated table, the wizard types an appropriate system response and the dialogue continues.

In order to enable large-scale parallel data collection and avoid the distracting latencies inherent in conventional WOZ scenarios (Bohus and Rudnicky, 2008), users and wizards are asked to contribute just a single turn to each dialogue. To ensure coherence and consistency, users and wizards must review all previous turns in that dialogue before they contribute their turns. Thus dialogues progress in a pipe-line. Many dialogues can be active in parallel and no worker ever has to wait for a response from the other party in the dialogue. Despite the fact that multiple workers contribute to each dialogue, we observe that dialogues are generally coherent yet diverse. Furthermore, this turn-level data collection strategy seems to encourage workers to learn and correct each other based on previous turns.

In this paper, the system was designed to assist users to find a restaurant in the Cambridge, UK area. There are three informable slots (*food, pricerange, area*) that users can use to constrain the search and six requestable slots (*address, phone, postcode* plus the three informable slots) that the user can ask a value for once a restaurant has been offered. There are 99 restaurants in the DB. Based on this domain, we ran 3000 HITs (Human Intelligence Tasks) in total for roughly 3 days and collected 1500 dialogue turns. After cleaning the data, we have approximately 680 dialogues in total (some of them are unfinished). The total cost for collecting the dataset was ~ 400 USD.

4 Empirical Experiments

Training Training is divided into two phases. Firstly the belief tracker parameters θ_b are

Table 1: Tracker performance in terms of Precision, Recall, and F-1 score.

Tracker type	Informable			Requestable		
	Prec.	Recall	F-1	Prec.	Recall	F-1
cnn	99.77%	96.09%	97.89%	98.66%	93.79%	96.16%
ngram	99.34%	94.42%	96.82%	98.56%	90.14%	94.16%

trained using the cross entropy errors between tracker labels \mathbf{y}_s^t and predictions \mathbf{p}_s^t , $L_1(\theta_b) = -\sum_t \sum_s (\mathbf{y}_s^t)^\top \log \mathbf{p}_s^t$. For the full model, we have three informable trackers (*food*, *pricerange*, *area*) and seven requestable trackers (*address*, *phone*, *postcode*, *name*, plus the three informable slots).

Having fixed the tracker parameters, the remaining parts of the model $\theta_{\setminus b}$ are trained using the cross entropy errors from the generation network language model, $L_2(\theta_{\setminus b}) = -\sum_t \sum_j (\mathbf{y}_j^t)^\top \log \mathbf{p}_j^t$, where \mathbf{y}_j^t and \mathbf{p}_j^t are output token targets and predictions respectively, at turn t of output step j . We treated each dialogue as a batch and used stochastic gradient descent with a small l_2 regularisation term to train the model. The collected corpus was partitioned into a training, validation, and testing sets in the ratio 3:1:1. Early stopping was implemented based on the validation set for regularisation and gradient clipping was set to 1. All the hidden layer sizes were set to 50, and all the weights were randomly initialised between -0.3 and 0.3 including word embeddings. The vocabulary size is around 500 for both input and output, in which rare words and words that can be delexicalised are removed. We used three convolutional layers for all the CNNs in the work and all the filter sizes were set to 3. Pooling operations were only applied after the final convolution layer.

Decoding In order to decode without length bias, we decoded each system response m_t based on the average log probability of tokens,

$$m_t^* = \operatorname{argmax}_{m_t} \{ \log p(m_t | \theta, u_t) / J_t \} \quad (13)$$

where θ are the model parameters, u_t is the user input, and J_t is the length of the machine response.

As a contrast, we also investigated the MMI criterion (Li et al., 2016) to increase diversity and put additional scores on delexicalised tokens to encourage task completion. This *weighted* decoding strategy has the following objective function,

$$m_t^* = \operatorname{argmax}_{m_t} \{ \log p(m_t | \theta, u_t) / J_t - \lambda \log p(m_t) / J_t + \gamma R_t \} \quad (14)$$

where λ and γ are weights selected on validation set and $\log p(m_t)$ can be modelled by a standalone LSTM language model. We used a simple heuristic for the scoring function R_t designed to reward giving appropriate information and penalise spuriously providing unsolicited information⁷. We applied beam search with a beamwidth equal to 10, the search stops when an end of sentence token is generated. In order to obtain language variability from the deployed model we ran decoding until we obtained 5 candidates and randomly sampled one as the system response.

Tracker performance Table 1 shows the evaluation of the trackers’ performance. Due to delexicalisation, both CNN type trackers and N-gram type trackers (Henderson et al., 2014) achieve high precision, but the N-gram tracker has worse recall. This result suggests that compared to simple N-grams, CNN type trackers can better generalise to sentences with long distance dependencies and more complex syntactic structures.

Corpus-based evaluation We evaluated the end-to-end system by first performing a corpus-based evaluation in which the model is used to predict each system response in the held-out test set. Three evaluation metrics were used: BLEU score (on top-1 and top-5 candidates) (Papineni et al., 2002), entity matching rate and objective task success rate (Su et al., 2015). We calculated the entity matching rate by determining whether the actual selected entity at the end of each dialogue matches the task that was specified to the user. The dialogue is then marked as successful if both (1) the offered entity matches, and (2) the system answered all the associated information requests (e.g. *what is the address?*) from the user. We computed the BLEU scores on the template-like output sentences before lexicalising with the entity value substitution.

⁷We give an additional reward if a requestable slot (e.g. address) is requested and its corresponding delexicalised slot or value token (e.g. <v.address> and <s.address>) is generated. We give an additional penalty if an informable slot is never mentioned (e.g. food=None) but its corresponding delexicalised value token is generated (e.g. <v.food>). For more details on scoring, please see Table 5.

Table 2: Performance comparison of different model architectures based on a corpus-based evaluation.

Encoder	Tracker	Decoder	Match(%)	Success(%)	T5-BLEU	T1-BLEU
Baseline						
lstm	-	lstm	-	-	0.1650	0.1718
lstm	turn recurrence	lstm	-	-	0.1813	0.1861
Variant						
lstm	rnn-cnn, w/o req.	lstm	89.70	30.60	0.1769	0.1799
cnn	rnn-cnn	lstm	88.82	58.52	0.2354	0.2429
Full model w/ different decoding strategy						
lstm	rnn-cnn	lstm	86.34	75.16	0.2184	0.2313
lstm	rnn-cnn	+ weighted	86.04	78.40	0.2222	0.2280
lstm	rnn-cnn	+ att.	90.88	80.02	0.2286	0.2388
lstm	rnn-cnn	+ att. + weighted	90.88	83.82	0.2304	0.2369

Table 2 shows the result of the corpus-based evaluation averaging over 5 randomly initialised networks. The *Baseline* block shows two baseline models: the first is a simple turn-level sequence to sequence model (Sutskever et al., 2014) while the second one introduces an additional recurrence to model the dependency on the dialogue history following Serban et al (Serban et al., 2015b). As can be seen, incorporation of the recurrence improves the BLEU score. However, baseline task success and matching rates cannot be computed since the models do not make any provision for a database.

The *Variant* block of Table 2 shows two variants of the proposed end-to-end model. For the first one, no requestable trackers were used, only informable trackers. Hence, the burden of modelling user requests falls on the intent network alone. We found that without explicitly modelling user requests, the model performs very poorly on task completion ($\sim 30\%$), even though it can offer the correct entity most of the time ($\sim 90\%$). More data may help here; however, we found that the incorporation of an explicit internal semantic representation in the full model (shown below) is more efficient and extremely effective. For the second variant, the LSTM intent network is replaced by a CNN. This achieves a very competitive BLEU score but task success is still quite poor ($\sim 58\%$ success). We think this is because the CNN encodes the intent by capturing several local features but lacks the global view of the sentence, which may easily result in an unexpected overfit.

The *Full model* block shows the performance of the proposed model with different decoding strategies. The first row shows the result of decoding using the average likelihood term (Equation 13) while the second row uses the *weighted* decoding strategy (Equation 14). As can be seen, the *weighted*

decoding strategy does not provide a significant improvement in BLEU score but it does greatly improve task success rate ($\sim 3\%$). The R_t term contributes the most to this improvement because it injects additional task-specific information during decoding. Despite this, the most effective and elegant way to improve the performance is to use the attention-based mechanism (*+att.*) to dynamically aggregate the tracker beliefs (Section 2.4). It gives a slight improvement in BLEU score (~ 0.01) and a big gain on task success ($\sim 5\%$). Finally, we can improve further by incorporating *weighted* decoding with the attention models (*+ att. + weighted*).

As an aside, we used t-SNE (der Maaten and Hinton, 2008) to produce a reduced dimension view of the action embeddings \mathbf{o}_t , plotted and labelled by the first three generated output words (full model w/o attention). The figure is shown as Figure 3. We can see clear clusters based on the system intent types, even though we did not explicitly model them using dialogue acts.

Human evaluation In order to assess operational performance, we tested our model using paid subjects recruited via Amazon Mechanical Turk. Each judge was asked to follow a given task and to rate the model’s performance. We assessed the subjective success rate, and the perceived comprehension ability and naturalness of response on a scale of 1 to 5. The full model with attention and weighted decoding was used and the system was tested on a total of 245 dialogues. As can be seen in Table 3, the average subjective success rate was 98%, which means the system was able to complete the majority of tasks. Moreover, the comprehension ability and naturalness scores both averaged more than 4 out of 5. (See Appendix for some sample dialogues in this trial.)

We also ran comparisons between the NN model

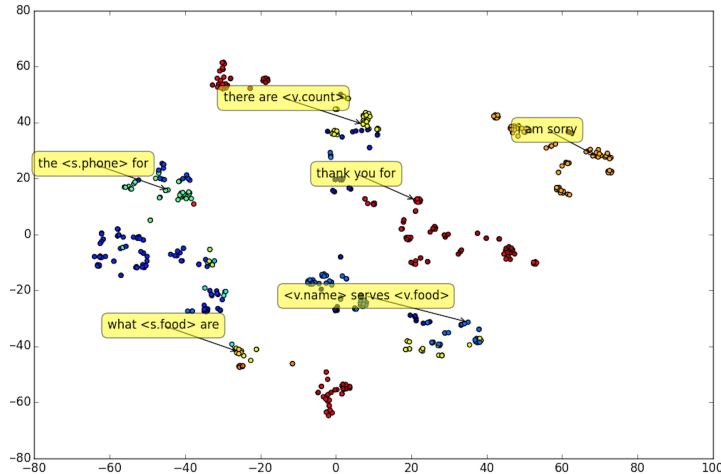


Figure 3: The action vector embedding \mathbf{o}_t generated by the NN model w/o attention. Each cluster is labelled with the first three words the embedding generated.

Table 3: Human assessment of the NN system. The rating for comprehension/naturalness are both out of 5.

Metric	NN
Success	98%
Comprehension	4.11
Naturalness	4.05
# of dialogues:	245

and a handcrafted, modular baseline system (*HDC*) consisting of a handcrafted semantic parser, rule-based policy and belief tracker, and a template-based generator. The result can be seen in Table 4. The *HDC* system achieved $\sim 95\%$ task success rate, which suggests that it is a strong baseline even though most of the components were hand-engineered. Over the 164 dialogues tested, the NN system (*NN*) was considered better than the handcrafted system (*HDC*) on all the metrics compared. Although both systems achieved similar success rates, the NN system (*NN*) was more efficient and provided a more engaging conversation (lower turn number and higher preference). Moreover, the comprehension ability and naturalness of the NN system were also rated higher, which suggests that the learned system was perceived as being more natural than the hand-designed system.

5 Conclusions and Future Work

This paper has presented a novel neural network-based framework for task-oriented dialogue systems. The model is end-to-end trainable using two

Table 4: A comparison of the NN system with a rule-based modular system (*HDC*).

Metric	NN	HDC	Tie
Subj. Success	96.95%	95.12%	-
Avg. # of Turn	3.95	4.54	-
Comparisons(%)			
Naturalness	46.95*	25.61	27.44
Comprehension	45.12*	21.95	32.93
Preference	50.00*	24.39	25.61
Performance	43.90*	25.61	30.49

* $p < 0.005$, # of comparisons: 164

supervision signals and a modest corpus of training data. The paper has also presented a novel crowd-sourced data collection framework inspired by the Wizard-of-Oz paradigm. We demonstrated that the pipe-lined parallel organisation of this collection framework enables good quality task-oriented dialogue data to be collected quickly at modest cost.

The experimental assessment of the NN dialogue system showed that the learned model can interact efficiently and naturally with human subjects to complete an application-specific task. To the best of our knowledge, this is the first end-to-end NN-based model that can conduct meaningful dialogues in a task-oriented application.

However, there is still much work left to do. Our current model is a text-based dialogue system, which can not directly handle noisy speech recognition inputs nor can it ask the user for confirmation when it is uncertain. Indeed, the extent to which this type of model can be scaled to much larger and wider domains remains an open question which we hope to pursue in our further work.

Wizard-of-Oz data collection websites

Task 02004: You are looking for and it should serve **gastropub food**. You don't care about the **price range**. You want to know the **address**.

Info Desk : Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area, price range or food type . How may I help you ?
 Customer : i want a gastropub food
 Info Desk : There are 4 restaurants serving gastropub food, what price range do you want ?
 Customer : i dont care

Next turn

Customer : (Your response)

i dont care about the price range, just give me the address please

Submit the HIT

Figure 4: The user webpage. The worker who plays a user is given a task to follow. For each mturk HIT, he/she needs to type in an appropriate sentence to carry on the dialogue by looking at both the task description and the dialogue history.

Info Desk : Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area, price range or food type . How may I help you ?
 Customer : i want a gastropub food
 Info Desk : There are 4 restaurants serving gastropub food, what price range do you want ?
 Customer : i dont care

Next turn

Please modify the following answers based on the latest customer response:

- What does user want?
 What is the **food type** the user wants? gastropub
 What is the **area** the user wants? not mentioned
 What is the **price range** the user wants? dont care
- What does user ask?
 Is the user asking for **food type** of an offered venue? No
 Is the user asking for **price range** of an offered venue? No
 Is the user asking for **area** of an offered venue? No
 Is the user asking for **postcode** of an offered venue? No
 Is the user asking for **phone number** of an offered venue? No
 Is the user asking for **address** of an offered venue? No
 Is the user mentioning any **restaurant names**? No

Finish

Info Desk : (Your response)

i would recommend backstreet bistro, a great gastropub restaurant in the centre, do you want their phone number ? end-of-dialogue

Submit the HIT

Name	Food	Area	Price Range	Phone	Address	Postcode
backstreet bistro	gastropub	centre	expensive	01223 306306	2 Slurton Street City Centre	C.B 1, 2 Q.A
royal standard	gastropub	east	expensive	01223 247877	290 Mill Road City Centre	C.B 1, 3 N.L
the cow pizza kitchen and bar	gastropub	centre	moderate	01223 308871	Corn Exchange Street	C.B 2, 3 Q.F
the slug and lettuce	gastropub	centre	expensive	--	34 - 35 Green Street	C.B 2, 3 J.U
nil	gastropub	nil	nil	nil	nil	nil

Showing 1 to 4 of 4 entries (filtered from 110 total entries)

Previous 1 Next

Figure 5: The wizard page. The wizard's job is slightly more complex: the worker needs to go through the dialogue history, fill in the form (top green) by interpreting the user input at this turn, and type in an appropriate response based on the history and the DB result (bottom green). The DB search result is updated when the form is submitted. The form can be divided into informable slots (top) and requestable slots (bottom), which contains all the labels we need to train the trackers.

Scoring Table

Table 5: Additional R_t term for delexicalised tokens when using weighted decoding (Equation 14). *Not observed* means the corresponding tracker has a highest probability on either *not mentioned* or *dontcare* value, while *observed* mean the highest probability is on one of the categorical values. A positive score encourages the generation of that token while a negative score discourages it.

Delexicalised token	Examples	R_t (observed)	R_t (not observed)
informable slot token	<s.food>, <s.area>,...	0.0	0.0
informable value token	<v.food>, <v.area>,...	+0.05	-0.5
requestable slot token	<s.phone>,<s.address>,...	+0.2	0.0
requestable value token	<v.phone>,<v.address>,...	+0.2	0.0

Acknowledgements

Tsung-Hsien Wen and David Vandyke are supported by Toshiba Research Europe Ltd, Cambridge. The authors would like to thank Ryan Lowe and Lukáš Žilka for their valuable comments.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint:1409.0473*.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, Seattle, Washington, USA. ACL.
- Dan Bohus and Alexander I. Rudnicky, 2008. *Sorry, I Didn't Catch That!*, pages 123–154. Springer Netherlands, Dordrecht.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, Doha, Qatar, October. ACL.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *JMLR*.
- Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *ICASSP*, pages 8367–8371, May.
- Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*, pages 292–299, Philadelphia, PA, USA, June. ACL.
- Matthew Henderson. 2015. Machine learning for dialog state tracking: A review. In *Machine Learning in Spoken Language Processing Workshop*.
- Karl Moritz Hermann, Tomáš Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701, Montreal, Canada. MIT Press.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November.
- Michael I. Jordan. 1989. Serial order: A parallel, distributed processing approach. In *Advances in Connectionist Theory: Speech*. Lawrence Erlbaum Associates.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665, Baltimore, Maryland, June. ACL.
- John F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transaction on Information Systems*.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, Doha, Qatar, October. ACL.
- Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*, pages 110–119, San Diego, California, June. ACL.
- Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *ACL*, pages 599–609, Berlin, Germany, August. ACL.
- Tomáš Mikolov, Martin Karafiat, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, pages 1045–1048, Makuhari, Japan. ISCA.
- Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *ACL*, pages 794–799, Beijing, China, July. ACL.
- Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint:1606.03777*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318, Stroudsburg, PA, USA. ACL.
- Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2015a. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint:1512.05742*.
- Iulian Vlad Serban, Alessandro Sordani, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015b. Hierarchical neural network generative models for movie dialogues. *arXiv preprint:1507.04808*.
- Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586, Beijing, China, July. ACL.
- Pei-Hao Su, David Vandyke, Milica Gasic, Dongho Kim, Nikola Mrksic, Tsung-Hsien Wen, and Steve J. Young. 2015. Learning from real users: rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Interspeech*, pages 2007–2011, Dresden, Germany. ISCA.

- Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *ACL*, pages 2431–2441, Berlin, Germany, August. ACL.
- Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*, pages 2440–2448. Curran Associates, Inc., Montreal, Canada.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, Montreal, Canada. MIT Press.
- David R. Traum, 1999. *Foundations of Rational Agency*, chapter Speech Acts for Dialogue Agents. Springer.
- Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*, Lille, France.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*, pages 2692–2700, Montreal, Canada. Curran Associates, Inc.
- Tsung-Hsien Wen, Aaron Heidele, Hung yi Lee, Yu Tsao, and Lin-Shan Lee. 2013. Recurrent neural network based language model personalization by social network crowdsourcing. In *Interspeech*, pages 2007–2011, Lyon France. ISCA.
- Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGdial*, pages 275–284, Prague, Czech Republic, September. ACL.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, pages 1711–1721, Lisbon, Portugal, September. ACL.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *NAACL-HLT*, pages 120–129, San Diego, California, June. ACL.
- Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *IEEE SLT*, pages 189–194, December.
- Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint:1512.00965*.
- Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer, Speech and Language*.
- Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Appendix: Sample dialogues

Table 6: Some samples of real conversational logs between online judges and the end-to-end system.

Human	: Restaurant in any area that serves Vietnamese food
Machine	: thanh binh serves vietnamese food type , is there anything else i can help you with ?
Human	: What is the phone number and post code
Machine	: the number is 01223 362456 and the area code is c.b 3, 0 a.f . is there anything else i can do for you ?
Human	: Good bye
Machine	: thank you for using the cambridge restaurant system .
Human	: i am looking for moderately priced halal food
Machine	: i am sorry , there are no halal restaurants in the moderate price range . would you like something else ?
Human	: how about european food
Machine	: galleria serves european food type and is in the moderate price range .
Human	: address and postcode
Machine	: the address is 33 bridge street and the area code is c.b 2, 1 u.w . is there anything else i can help you with today ?
Human	: thanks
Machine	: galleria is a european restaurant in the moderate price range . would you like their phone ?
Human	: thank you
Machine	: thank you for using the cambridge restaurant system . good bye
Human	: Hi, I'm hungry for some Indonesian food
Machine	: i am sorry , there are no indonesian restaurants . would you like a different food ?
Human	: Oh no. How about Indian?
Machine	: there are 9 restaurants serving indian types of food . do you have a preference on side of town ?
Human	: East side please
Machine	: sitar tandoori is a indian restaurant in the east side of town . would you like their information ?
Human	: Yes, the address and phone number please
Machine	: sitar tandoori is located at 43 high street cherry hinton cherry hinton and their phone number is 01223 249955 .
Human	: Okay, thank you
Machine	: thank you , goodbye .
Human	: hello
Machine	: what cuisine would you like ?
Human	: afghan
Machine	: i am sorry , there are no afghan restaurants . would you like something else ?
Human	: korean
Machine	: little seoul serves korean food , it is located at 108 regent street city centre .
Human	: is it expensive?
Machine	: little seoul serves korean food and is in the expensive price range .
Human	: what area is that
Machine	: little seoul is located in the centre area of town . would you like their number ?
Human	: yes
Machine	: the telephone number is 01223 308681 . is there anything else i can help you with today ?
Human	: no goodbye
Machine	: thank you for using the cambridge restaurant system , goodbye .
Human	: I want to find a restaurant in the east part of town that serves creative food
Machine	: i am sorry , there are no creative restaurants in the east side of town . would you like a different restaurant ?
Human	: how about indian food?
Machine	: curry prince is a indian restaurant in the east of town . would you like their phone number ?
Human	: yes, may i have the address and phone number?
Machine	: curry prince is located at 451 newmarket road fen ditton and their phone is 01223 566388 . is there anything else i can help you with today ?
Human	: no, thank you goodbye
Machine	: thank you for using the cambridge restaurant system . good bye
Machine	: what food are you interested ?
Human	: english
Machine	: i am sorry , there are no english restaurants . would you like to try a different food type ?
Human	: chinese
Machine	: there are 8 restaurants serving chinese type of cuisine . do you have a preference of area ?
Human	: south part
Machine	: the good luck chinese food takeaway is a chinese restaurant in the south of town . would you like the phone number ?
Human	: no. just address and price range
Machine	: the good luck chinese food takeaway is located at 82 cherry hinton road cherry hinton . is there anything else i can help you with ?
Human	: what is their price range?
Machine	: the good luck chinese food takeaway is expensively priced chinese restaurant in the south area . would you like more information ?
Human	: no goodbye
Machine	: thank you for using the cambridge restaurant system . good bye .
Human	: Is there a gastropub anywhere around here?
Machine	: there are 3 gastropub restaurants . would you like more information ?
Human	: Is there one on the west side?
Machine	: i am sorry , there are no gastropub restaurants in the west side of town . would you like something else ?
Human	: How about east?
Machine	: royal standard is a gastropub restaurant in the east of town .
Human	: What's the address?
Machine	: the address is 290 mill road city centre .
Human	: Thanks, bye
Machine	: thank you for using the cambridge restaurant system . good bye

May I take your order? A Neural Model for Extracting Structured Information from Conversations

Baolin Peng¹, Michael L. Seltzer²,
Yun-Cheng Ju², Geoffrey Zweig² and Kam-Fai Wong¹

¹The Chinese University of Hong Kong, Shatin, N.T. Hong Kong

²Microsoft Research, One Microsoft Way, Redmond, WA, USA

{blpeng, kfwong}@se.cuhk.edu.hk
{mseltzer, yuncj, gzweig}@microsoft.com

Abstract

In this paper we tackle a unique and important problem of extracting a structured order from the conversation a customer has with an order taker at a restaurant. This is motivated by an actual system under development to assist in the order taking process. We develop a sequence-to-sequence model that is able to map from unstructured conversational input to the structured form that is conveyed to the kitchen and appears on the customer receipt. This problem is critically different from other tasks like machine translation where sequence-to-sequence models have been used: the input includes two sides of a conversation; the output is highly structured; and logical manipulations must be performed, for example when the customer changes his mind while ordering. We present a novel sequence-to-sequence model that incorporates a special attention-memory gating mechanism and conversational role markers. The proposed model improves performance over both a phrase-based machine translation approach and a standard sequence-to-sequence model.

1 Introduction

Extracting structured information from unstructured text is a critically important problem in natural language processing. In this paper, we attack a deceptively simple form of the problem: understanding what a customer wants when ordering at a restaurant. In this problem, we seek to convert the conversation between the customer and the order taker, i.e. the waiter or waitress, into the structured form that is conveyed to the kitchen to prepare the food, and which appears on the customer receipt.

Waiter: Hi, how can I help you ?
Customer: We'd like a large cheese pizza.
Waiter: Any toppings?
Customer: Yeah, how about pepperoni and two diet cokes.
Waiter: What size?
Customer: Uh, medium and make that three cokes.
Waiter: Anything else?
Customer: A small Caesar salad with the dressing on the side
Waiter: Sure, is that it?
Customer: Yes, that's all, thanks.

Figure 1: A conversation example of an order-taking interaction at a restaurant.

Item	Size	Qty	Modifiers
Pizza	large	1	add pepperoni
Caesar Salad	small	1	side dressing
Diet Coke	medium	3	

Table 1: An example of the structured data record corresponding to the conversation in Figure 1

We develop this system to analyze real-time interactions with the aim of discovering errors in the order-entry process. Note that the objective is to analyze the interaction and suggest corrections to the human order-taker. Thus, we take both sides of the order-taking interaction as input, and are not attempting to predict the order-taker's side of the conversation.

While we focus on the restaurant domain in this work, this problem is relevant in any scenario in which a conversation results in the creation of structured information. Other examples include a sales interaction which results in a purchase order, a call to a help desk which results in a service record, or a conversation with a travel agent that results in an itinerary.

An example of the problem of interest is shown in Figure 1. The structured data record that corresponds to this conversation is shown in Table 1. There are several things to note about this example:

- The output is a stylized and structured representation of the input
- The items in the structured order may appear in a different sequence than they are mentioned
- Inference occurs across turns, for example that “medium” applies to the coke and not the pizza whose size was earlier specified
- Logical manipulations must be done, for example changing the number of cokes from two to three
- In contrast to machine translation, we do not wish to create a verbatim “translation” of the input, but instead a logical distillation of it

To attack this problem, we implemented two baselines and several sequence-to-sequence models. The first baseline is an information-retrieval approach based on a TF-IDF match (Salton et al., 1975) which finds the most similar conversation in the training data, and returns the associated order. The second uses phrase-based machine translation (Koehn et al., 2003) to “translate” from the conversational input to the tokens in the structured order. We compare these to a sequence-to-sequence (s2s) model with attention (Chan et al., 2016; Bahdanau et al., 2014; Devlin et al., 2015; Yao and Zweig, 2015; Sutskever et al., 2014; Mei et al., 2016), and then extend the s2s model with the addition of a gating mechanism on the attention memory and with an auxiliary input that indicates the conversational role of the speaker (customer or order-taker). We show that it is in fact possible to extract the orders from conversations recorded at a real restaurant ¹, and achieve an F measure of over 70 from raw text and 65 from ASR transcriptions.

2 Problem Formulation

The precise problem setting in this paper is as follows. The training data consists of input/output pairs of examples $(X_1, Y_1), \dots, (X_N, Y_N)$, where X_k is a conversation consisting of several utterances, similar to the example shown in Figure 1, and Y_k is the corresponding structured data record such as the one in Table 1.

¹The restaurant will remain anonymous for business reasons, and we have changed the names of menu items in our examples accordingly.

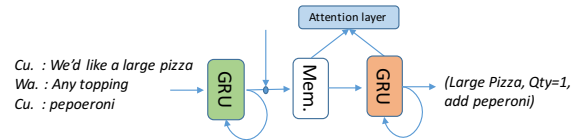


Figure 2: An input unstructured conversation and the corresponding structured record.

Given a conversation X_k , the goal of our model is to extract the structured data record Y_k so that:

$$Y_k = \operatorname{argmax}_Y \log P(Y|X_k) \quad (1)$$

We cast this task as a sequence modeling problem which aims to map the sequence of words in a conversation X_k to the sequence of tokens in the corresponding structured data record Y_k . The input sequence is formed by concatenating the utterances in the conversation, while the output sequence is formed by concatenating the rows in the structured data record. For example, the utterances in the conversation shown in Figure 1 are concatenated to predict the sequence $y = \text{Pizza, size=large, qty=1, modifiers=(add peperoni)} \mid \text{Diet Coke, size=medium, qty=3} \mid \text{Caesar Salad, size=small, qty=1, modifiers=(side dressing)}$ which is derived from Table 1. Under this sequential model, the conditional probability of the structured data record Y given the observed conversation X can be written as

$$P(Y|X, \theta) = \prod_{t=1}^T P(y_t|y_{1:t-1}, X, \theta) \quad (2)$$

where $y_{1:t-1}$ denotes the first $t - 1$ terms in the structured data record and θ represents the model parameters.

3 Model

The proposed model is based on an encoder-decoder architecture with attention (Bahdanau et al., 2014), as shown in Figure 2. The encoder network reads the input conversation \mathbf{X} one word at a time and updates its hidden state h_t according to current input w_t and previous hidden state h_{t-1} ,

$$h_t = f_e(w_t, h_{t-1}), t \in \{1, \dots, M\} \quad (3)$$

where f_e is a nonlinear function which is elaborated in the following section. After reading all the tokens, the encoder network yields a context

vector c as the representation of the entire conversation.

The decoder then processes this representation and generates a hypothesized structured data record Y as an output sequence, word by word given the context vector c and all previous predicted tokens. The conditional probability can be expressed as follows:

$$P(y_t|y_1, \dots, y_{t-1}, \mathbf{X}) = f_d(y_{t-1}, s_t, c) \quad (4)$$

$$s_t = g(y_{t-1}, s_{t-1}, c) \quad t \in \{1, \dots, N\} \quad (5)$$

where f_d and g are nonlinear functions and s_t is the hidden state of decoder at time t . Critically, our decoder also utilizes an attention mechanism, which stores the intermediate encoder representations of each input word for use by the decoder.

Two improvements to the conventional encoder-decoder model architecture are proposed in this work. First, we incorporate gates controlled by the encoder into the neural attention memory to adaptively modulate the representations in the memory based on their semantic importance. Second, we propose a way to incorporate conversational role information into the model to reflect the fact that different participants in a multi-party interaction have different roles and the meaning of certain utterances may be dependent on the speaker's role.

A detailed illustration of the proposed model is shown in Figure 3. We elaborate on each component of this model in the following sections.

3.1 Encoder Network

The encoder network is designed to generate a semantically meaningful representation of unstructured conversations. Several neural network architectures have been proposed for this purpose, including CNNs (Kalchbrenner et al., 2014; Hu et al., 2014), RNNs (Sutskever et al., 2014) and LSTMs (Hochreiter and Schmidhuber, 1997). In this work, we use an encoder constructed from a recurrent neural network with gated RNN units (GRU) (Cho et al., 2014). The GRU has been shown to alleviate the gradient vanishing problem of RNNs, enabling the model to learn long term dependencies in the input sequence. GRUs have been shown to perform comparably to LSTMs (Chung et al., 2014).

At time t , the new state of a GRU is computed as follows:

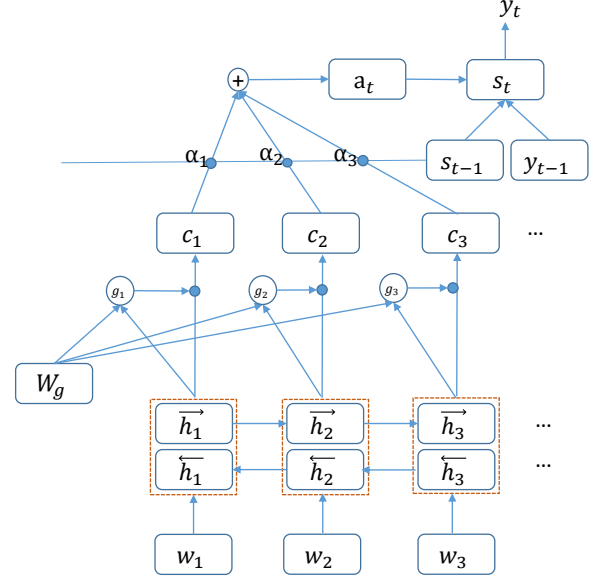


Figure 3: Graphical structure of memory-gated encoder-decoder model with attention mechanism. w_1 represents input; \vec{h}_1 and \overleftarrow{h}_1 are the hidden states of forward and backward GRUs, respectively. g_1, α_1 represent the context gates and attention weights, respectively. Small dot node means element-wise product.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (6)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (7)$$

$$\hat{h}_t = \tanh(W_h x_t + U_h (r_t \odot h_{t-1})) \quad (8)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \quad (9)$$

where \odot stands for element-wise multiplication. W, U are weight matrixes applied to input and previous hidden state, respectively. h_t is a linear combination of the previous state h_{t-1} and the hypothesis state \hat{h}_t . \hat{h}_t is computed with new sequence information. The update gate, z_t , controls to what extent the past information is kept and how much new information is added. The reset gate, r_t , controls to what extent the history state contributes to the hypothesis state. If r_t is zero, then GRU ignores all the history information.

The conversation encoding is obtained by concatenating the GRU hidden state vectors from the forward and backward directions. Thus the encoder operation can be summarized as follows

$$x_t = W_e w_t, t \in [1, T] \quad (10)$$

$$\vec{h}_t = \overrightarrow{GRU}(x_t), t \in [1, T] \quad (11)$$

$$\overleftarrow{h}_t = \overleftarrow{GRU}(x_t), t \in [T, 1] \quad (12)$$

$$h_t^+ = \vec{h}_t \oplus \overleftarrow{h}_t \quad (13)$$

where w_t is the one-hot input vector, W_e is the embedding matrix, and x_t is the word embedding for w_t . The functions $\overrightarrow{GRU}(x_t)$ and $\overleftarrow{GRU}(x_t)$ represent the GRU operating in the forward and backward directions, respectively, with processing defined by equations 6–9.

This produces a sequence of context vectors, h_t^+ which are subsequently consumed by an attention mechanism in the decoder. We use the final attention vector h_T^+ to initialize the hidden state of the decoder.

3.2 Memory Gate

In most sequence-to-sequence tasks such as machine translation, every word in the input is important. However, in our scenario, where the input to the system is conversational speech, not all the words in the conversation contribute to the prediction of structured data record. For example, it is reasonable to ignore the chit-chat that is present in many conversations. Further, in other tasks, gating mechanisms have been shown to be useful to dynamically select important information (Yao et al., 2015; Hochreiter and Schmidhuber, 1997; Tu et al., 2016).

In light of this, we propose the use of an additional memory gate to select important information from the memory vector. The memory gate we use consists of a single-layer feed-forward neural network

$$g_t = \sigma(W_g h_t^+ + b_g) \quad (14)$$

where σ is a sigmoid activation function and W_g and b_g are weight matrix and bias, respectively, and h_t^+ is the context vector at time t defined in equation 10. The gate is then applied to the context vector h_t^+ using an element-wise multiplication operation.

$$c_t = g_t \odot h_t^+ \quad (15)$$

After applying memory gate, the gated context vector c_t is then fed into attention memory of the decoder network in place of the original context vector h_t^+ . Figure 4 illustrates an example of the gating weights for a sample utterance. The darker colors indicates values close to 1 while the lighter colors indicate values close to 0. As the figure shows, the network learns to suppress semantically unimportant words.

3.2.1 Role Information

In many sequence-to-sequence models, there is no notion of different speakers with different roles. Inspired by the work in dialog generation (Li et al., 2016) and spoken language understanding (Hori et al., 2016), we propose the addition of speaker information into the encoder network to explicitly model the interaction patterns of the customer and order-taker.

Specifically we learn separate word and role embeddings, and concatenate them to form the input. The input to the encoder network becomes:

$$x_t^w = W_e w_t, t \in [1, T] \quad (16)$$

$$x_t^r = W_r r_t, t \in [1, T] \quad (17)$$

$$x_t = x_t^w \oplus x_t^r, t \in [1, T] \quad (18)$$

3.3 Decoder Network

The decoder network is used to predict the next word given all the previously predicted words and the context vectors from the encoder network (Luong et al., 2015; Bahdanau et al., 2014).

We use an RNN with GRU units to predict each word y_t sequentially based on the previously predicted word y_{t-1} and the output of the attention process a_t that computes a weighted combination of the context vectors in memory.

If we define s_t as the hidden layer of the decoder at time t , the decoder’s operation can be expressed as

$$s_t = \overrightarrow{GRU}(y_{t-1} \oplus a_t) \quad (19)$$

$$y_t = \text{softmax}(W_o s_t + b_o) \quad (20)$$

where $y_{t-1} \oplus a_t$ is the concatenation of the previously predicted output y_{t-1} and the output of the attention process a_t , and $\overrightarrow{GRU}(\cdot)$ is defined by equations 6–9, as before.

The attention vector a_t is computed as a linear combination of the gated context vectors generated by the encoder network. This can be written as

$$a_t = \sum_{j=1}^M \alpha_{ij} c_j \quad (21)$$

where the weights α_{ij} are computed as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^N \exp(e_{ik})} \quad (22)$$

A single-layer feed-forward neural network is used to compute e_{ij} as

$$e_{ij} = V_a^T \tanh(W_a s_{t-1} + U_a c_j) \quad (23)$$

where V_a , W_a , and U_a are weight matrices.

3.4 Model Training

The model is trained to maximize the log probability of the structured data records given the corresponding conversation,

$$\sum_{(Y_k, X_k) \in D} \log P(Y_k | X_k) \quad (24)$$

where D is the set containing all the training pairs and $P(Y_k | X_k)$ is computed with equation 2. The standard adadelta algorithm (Zeiler, 2012) is used for parameter updates. Gradients are clipped to 1 to avoid exponentially increasing values (Pascanu et al., 2013).

4 Experiments

In this section, we evaluate our proposed model on two data sets and compare performance with several baseline systems.

4.1 Data sets

We conducted experiments on a corpus of conversations between a customer and an order taker (waiter or waitress) captured in a real restaurant environment. The conversations were manually transcribed by professional annotators. There are 4823 examples in the training set, 543 in the development (dev) set, and 843 in the test set. There are approximately 260 unique items in the record and 150 unique modifiers on these items, but not all modifiers apply to all items. We experimented with two version of the dev and test sets. The first is manually transcribed in the same manner as the training set, while the second is generated by a speech recognition decoder that was trained on the conversations in the training set. We denote the second set as *ASR-dev* and *ASR-test*. Table 2 lists the statistics of the data sets. Note that the audio of a conversation was collected as a single file and then automatically segmented into turns for ASR decoding. This process was not perfect and likely introduced some errors. Thus, the average length and number of turns of differ between the ASR transcriptions and the manual transcriptions.

Data Set	Avg Turns	Avg Length	Avg # Items
Training	8.8	53.5	3.7
Dev	9.7	57.07	3.8
Test	8.7	50.96	3.5
ASR-Dev	8.5	49.8	3.8
ASR-Test	7.8	44.7	3.5

Table 2: Statistics of the experimental corpus. The table denotes the average number of utterances in a conversation, average length of a conversation in words, and average number of items in an order.

4.2 Experimental setup

All words are lower-cased and an unknown word token is used for words which appear less than four times in the training set. The word embedding matrix is initialized by randomly sampling from a normal distribution, and scaled by 0.01. The recurrent connections of the GRU are initialized with orthogonal matrices (Saxe et al., 2013) and biases are initialized to zero. A single layer GRU is used for both the encoder and decoder. The network has 600 hidden units and uses 300-dimensional word embeddings. The dropout rate is set to 0.5. We did not tune hyper-parameters except for the dimension of the role embedding which is selected from $\{3, 5, 10\}$ on the dev set. During inference, we use beam search decoding with a beam of 5 to generate the structured records. In order to decode without a length bias, the log probability of decoded results is normalized by the number of tokens.

4.3 Evaluation

A typical metric to evaluate a generation system is BLEU score (Papineni et al., 2002) which uses ngram overlap to quantify the degree to which a hypothesis matches the reference. However, our scenario is more demanding: order items are either correct or incorrect. Therefore, we adopt precision and recall at the item level as our evaluation metric. Note that an item is defined as a row in the structured data record and typically includes multiple fields. Using Table 1 as an example, there are three items to be scored. Only when the model produces an item that is exactly the same as the reference item do we count it as correct. As an additional measure, we report accuracy of the entire order, in which every item in an order must be correct for the order to be counted as correct.

	Model		Dev			Test				
	Gate	Role	Recall	Prec	F1	Accy	Recall	Prec	F1	Accy
IR	-	-	26.5	22.7	24.5	11.4	29.7	25.6	27.5	14.1
PBMT	-	-	64.4	19.3	35.2	29.3	62.6	20.8	36.0	28.4
NAM	-	-	64.9	70.9	67.9	45.7	68.4	71.3	69.8	48.1
NAM	-	✓	65.6	71.6	68.6	45.3	68.8	72.9	70.8	49.1
NAM	✓	-	67.6	72.7	70.1	46.2	68.3	74.1	71.1	48.5
NAM	✓	✓	66.9	71.6	69.2	48.8	70.2	72.3	71.2	51.8

Table 3: Results of different methods on dev and test set. Human transcriptions are used.

	Model		Dev			Test				
	Gate	Role	Recall	Prec	F1	Accy	Recall	Prec	F1	Accy
IR	-	-	21.9	18.9	20.3	6.9	25.7	19.3	23.8	10.2
PBMT	-	-	56.8	20.4	34.1	23.3	56.9	21.5	35.0	24.7
NAM	-	-	56.7	63.5	60.0	36.9	60.3	66.7	63.4	40.6
NAM	-	✓	57.1	64.7	60.8	38.1	62.5	67.4	64.9	42.5
NAM	✓	-	57.0	64.6	60.7	39.2	60.3	68.3	64.2	40.8
NAM	✓	✓	58.5	65.2	61.8	40.5	63.0	68.4	65.7	45.9

Table 4: Results of different methods on ASR-dev and ASR-test set.

4.4 Baseline systems

We compare the performance of our neural model with baseline models that employ information retrieval (IR) and phrase-based machine translation (PBMT) approaches.

IR: The IR method treated the training set of transcriptions as a collection of documents, each mapped to a corresponding order. The test conversation was used as a query to find the most similar training set conversation. The corresponding order was returned as the estimated order. In our experiment, we use TFIDF to compute the similarity score.

PBMT: The goal of a phrase-based translation model is to map a conversation into its structured record with alignment and language models. In our experiments, we use the Moses decoder, a state-of-the-art phrase-based MT system available for research purposes. We use GIZA++ (Och and Ney, 2003) to learn word alignment and *irstlm* to learn the language model. The models are trained on the conversation/order pairs in the training set and used to predict the structured data record given a conversation.

4.5 Results

First we discuss the performance of our models on manually transcribed data and then examine the results on ASR recognized data. Table 3 lists the experiment results on manually transcribed dev and test sets. We refer to our model as the neural attention model (NAM). We see that the NAM is superior to both the IR and PBMT methods by a large margin. Both the proposed memory gate and role modifications yield improvements over the basic NAM. When combined, these produce the best performance in terms of accuracy on the dev set, and both F1 and accuracy on the test set. While there are only small differences in the scores among some of the NAM methods, we are unaware of a measure of statistical significance suitable for this task.

Though not reported, we also found that a basic encoder-decoder s2s model without attention performs poorly; it cannot summarize information across multiple turns into a single vector. The attention mechanism, acting on the entire encoding sequence, is critical in our task.

Table 4 shows results on the ASR-dev and ASR-test sets. These data sets are quite noisy since the speech recognizer in this domain has a word error

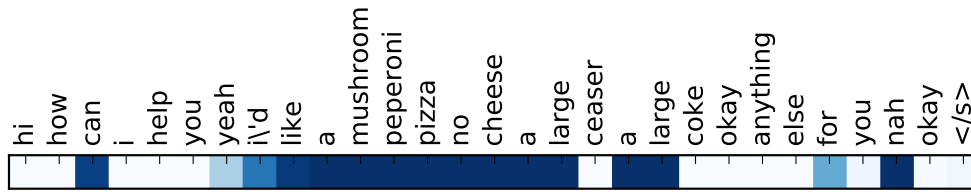
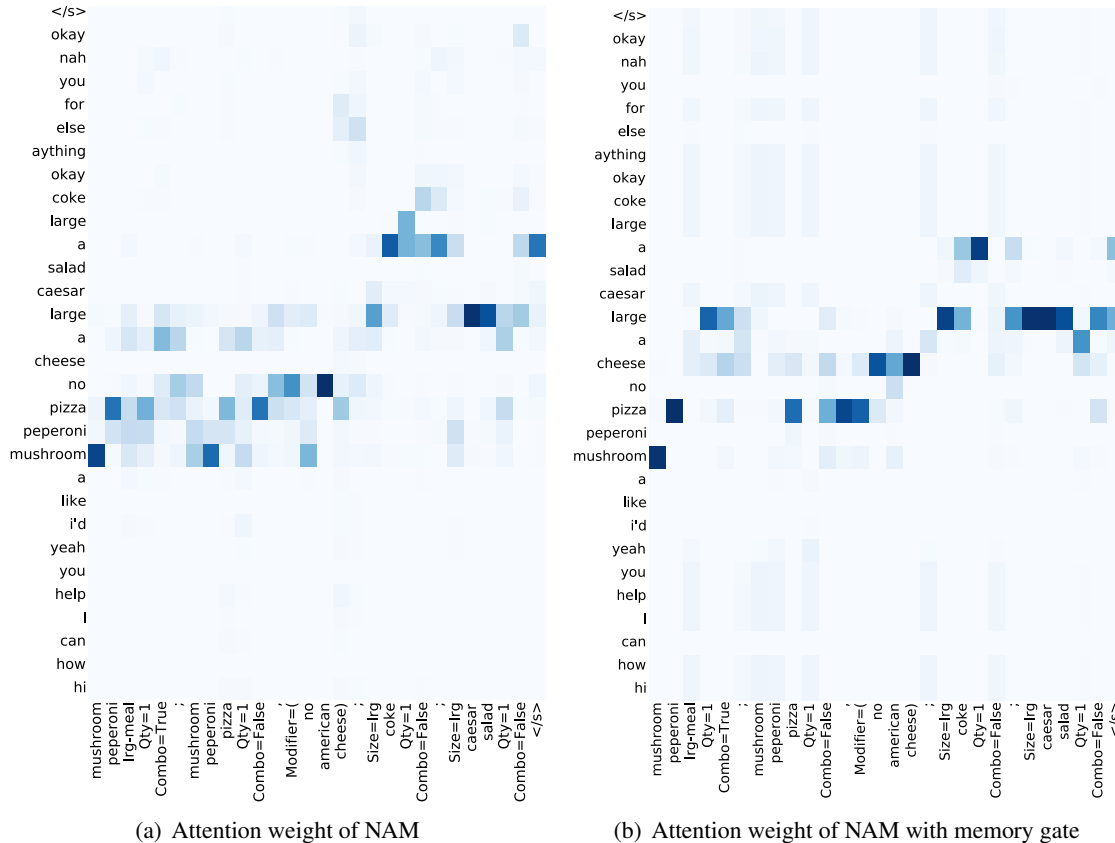


Figure 4: Example of memory gate weights at each time stamp.



(a) Attention weight of NAM

(b) Attention weight of NAM with memory gate

Figure 5: Examples of attention weights of models (a) without memory gate and (b) with memory gate. (b) shows sparse and more focused attention weights. (Better viewed in color.)

rate around 25%. With this noisy data, we find that the memory gate and role additions consistently improve performance. When combined, both F1 and accuracy improved.

4.6 Qualitative analysis

Figure 6 shows a sample input and the output from each model. We see that the NAM augmented with memory gates and role information successfully captures the interaction and generates the correct record.

To better understand the proposed model, we visualize the attention weight at each time step in Figure 5. The figure compares the attention weights produced by a conventional context mem-

ory and the proposed gated context memory. We see that both models are able to learn good soft alignment between the input conversation and the output structured data record. However, the attention weights in 5(b), with our proposed gated attention mechanism, are sparser than those in 5(a) and better able to ignore uninformative terms in the input.

5 Related Work

There has been much work on information extraction from single utterances. Kate and Mooney (2006) proposed the use of SVM classifiers based on string kernels to parse natural language to a formal meaning representation. Wong and

Waiter: My name's Alexis how can I help you.
Customer: Yeah can I get a cheese pizza combo?
Waiter: Something to drink?
Customer: Change that to a mushroom pizza
combo with a large sprite no ice.
Waiter: Okay.
Customer: Thank you.
Waiter: Thank you.

NAM:

Cheese Pizza, Qty=1, Combo=True
Sprite, Size=Lrg, Qty=1, Modifier=(no ice);
Caesar Salad, Size=Med, Qty=1

NAM + Memory Gate:

Cheese Pizza, Qty=1, Combo=True
Sprite, Size=Lrg, Qty=1, Modifier=(no ice);
Caesar Salad, Size=Med, Qty=1
Mushroom Pizza, Qty=1, Combo=True
Sprite, Size=Lrg, Qty=1, Modifier=(no ice);
Caesar Salad, Size=Med, Qty=1

NAM + Memory Gate + Role:

Mushroom Pizza, Qty=1, Combo=True
Sprite, Size=Lrg, Qty=1, Modifier=(no ice);
Caesar Salad, Size=Med, Qty=1

Figure 6: Examples of outputs generated by each model for the conversation in first row.

Mooney (2006) used syntax-based statistical machine translation method to do semantic parsing. Translation of natural language to a formal meaning representation is captured by a synchronous context-free grammar in (Wu, 1997; Chiang et al., 2006). Quirk et al. (2015) created models to map natural language descriptions to executable code using productions from the formal language. Beltagy and Quirk (2016) improved the performance of semantic parsing on If-Then statements by using neural networks to model derivation trees and leveraged several techniques like synthetic training data from paraphrases and grammar combinations to improve generalization and reduce overfitting. In addition, there are some other research works focusing on text generation from structured data records. Angeli et al. (2010) proposed of a domain independent probabilistic approach to performing content selection and surface realization, making text generation as a local decision process. Konstas and Lapata (2013) created a global model to generate text from structured records, which jointly modeled content selection and surface realization with a probabilistic context-free grammar. In contrast, in this paper we focus on generating structured data records from text descriptions.

Using spoken language understanding techniques, (Mesnil et al., 2015) tag each word in a sentence with a predefined slot. A dialog modeling approach (Young et al., 2013) is also relevant to our task. However, this approach requires the definition of semantic slot names and human labeling of dialog acts in each utterance.

There are a number of relevant applications of neural attention models. Nallapati et al. (2016) proposed using sequence to sequence model to summarize source code into natural language; they used a LSTM as encoder and another attentional LSTM and decoder to jointly learn content selection and realization. Dong and Lapata (2016) presented a sequence to sequence model with a tree structure decoder to map natural language to its logical form. The tree structure decoder shows superior performance on data that has nested output structure. It has also been used in other domains including machine translation (Sutskever et al., 2014; Bahdanau et al., 2014), and image caption generation (Fang et al., 2015). From this perspective, the most related work is (Mei et al., 2016) in which they proposed using a sequence-to-sequence model to map navigational instructions in natural language to actions, which is conceptually similar to our work. However, we start from conversations and our structured data records are more complex.

6 Conclusion

In this paper we have presented an end to end method for extracting structured information from unstructured conversations using an encoder-decoder neural network. The restaurant-ordering domain we study is distinguished from past work by its conversational nature, and the need to handle user corrections and modifications. We incorporate a memory gate and role information into the encoder network to selectively keep important information and capture interaction patterns between conversation participants. Experimental results on both a human transcribed data set and ASR-recognized data set demonstrate the feasibility and effectiveness of our approach.

Acknowledgements

This work was done while Baolin Peng was an intern at Microsoft Research.

References

- Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA, October. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.
- I. Beltagy and Chris Quirk. 2016. Improved semantic parsers for if-then statements. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 726–736, Berlin, Germany, August. Association for Computational Linguistics.
- William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 4960–4964. IEEE.
- David Chiang, Mona T. Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. 2006. Parsing arabic dialects. In Diana McCarthy and Shuly Wintner, editors, *EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy*. The Association for Computer Linguistics.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of NIPS Deep Learning and Representation Learning Workshop*.
- Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105, Beijing, China, July. Association for Computational Linguistics.
- Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany, August. Association for Computational Linguistics.
- Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1473–1482.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Chiori Hori, Takaaki Hori, Shinji Watanabe, and John R. Hershey. 2016. Context-sensitive and role-dependent spoken language understanding using bidirectional and attention lstms. In *Interspeech 2016*, pages 3236–3240.
- Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.
- Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 913–920, Sydney, Australia, July. Association for Computational Linguistics.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003*. The Association for Computational Linguistics.
- Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res. (JAIR)*, 48:305–346.
- Jiwei Li, Michel Galley, Chris Brockett, Georgios Spathourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany, August. Association for Computational Linguistics.

- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.
- Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2772–2778.
- Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Z. Hakkani-Tür, Xiaodong He, Larry P. Heck, Gökhan Tür, Dong Yu, and Geoffrey Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 23(3):530–539.
- Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.
- Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318.
- Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888, Beijing, China, July. Association for Computational Linguistics.
- Gerard Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.
- Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.
- Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2016. Context gates for neural machine translation. *CoRR*, abs/1608.06043.
- Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA, June. Association for Computational Linguistics.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.
- Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 3330–3334. ISCA.
- Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *CoRR*, abs/1510.08565.
- Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.
- Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

A Two-stage Sieve Approach for Quote Attribution

Grace Muzny¹, Michael Fang¹, Angel X. Chang^{1,2}, and Dan Jurafsky¹

¹Stanford University, Stanford, CA 94305

²Princeton University, Princeton, NJ 08544

{muzny, mjfang, angelx, jurafsky}@cs.stanford.edu

Abstract

We present a deterministic sieve-based system for attributing quotations in literary text and a new dataset: QuoteLi3¹. Quote attribution, determining who said what in a given text, is important for tasks like creating dialogue systems, and in newer areas like computational literary studies, where it creates opportunities to analyze novels at scale rather than only a few at a time. We release QuoteLi3, which contains more than 6,000 annotations linking quotes to speaker mentions and quotes to speaker entities, and introduce a new algorithm for quote attribution. Our two-stage algorithm first links quotes to mentions, then mentions to entities. Using two stages encapsulates difficult sub-problems and improves system performance. The modular design allows us to tune either for overall performance or for the high precision appropriate for many use cases. Our system achieves an average F-score of 87.5% across three novels, outperforming previous systems, and can be tuned for precision of 90.4% at a recall of 65.1%.

1 Introduction

Dialogue, representing linguistic and social relationships between characters, is an important component of literature. In this paper, we consider the task of quote attribution for literary text: identifying the speaker for each quote. This task is important for developing realistic character-specific conversational models (Vinyals and Le, 2015; Li et al., 2016), analyzing discourse structure, and literary studies (Muzny et al., 2016). But identifying speakers can be difficult; authors often refer to the

speaker only indirectly via anaphora, or even omit mention of the speaker entirely (Table 1).

Prior work has produced important datasets labeling quotes in novels, providing training data for supervised methods. But some of these model the quote-attribution task at the mention-level (Elson and McKeown, 2010; O’Keefe et al., 2012), and others at the entity-level (He et al., 2013), leading to labels that are inconsistent across datasets.

We propose entity-level quote attribution as the end goal but with mention-level quote attribution as an important intermediary step. Our first contribution is the QuoteLi3 dataset, a unified combination of data from Elson and McKeown (2010) and He et al. (2013) with the addition of more than 3,000 new labels from expert annotators. This dataset provides both mention and entity labels for *Pride and Prejudice*, *Emma*, and *The Steppe*.

Next, we describe a new deterministic system that models quote attribution as a two-step process that i) uses textual cues to identify the mention that corresponds to the speaker of a quote, and ii) resolves the mention to an entity. This system improves over previous work by 0.8-2.1 F1 points and its modular design makes it easy to add sieves and incorporate new knowledge.

In summary, our contributions are:

- A unified dataset with both quote-mention and quote-speaker links labeled by expert annotators.
- A new quote attribution strategy that improves on all previous algorithms and allows the incorporation of both rich linguistic features and machine learning components.
- A new annotation tool designed with the specifics of this task in mind.

We freely release the data, system, and annotation tool to the community.²

¹Quotes in Literary text from 3 novels.

²nlp.stanford.edu/~muzny/quoteli.html

Type	Example	Speaker
Explicit	“Do you really think so?” cried Elizabeth , brightening up ...	Elizabeth Bennet
Anaphoric (pronoun)	“You are uniformly charming!” cried he , with an air of awkward gallantry;	Mr. Collins
Anaphoric (other)	“I see your design, Bingley,” said his friend .	Mr. Darcy
Implicit	“Then, my dear, you may have the advantage of your friend, and introduce Mr. Bingley to her.”	Mr. Bennet
	“Impossible, Mr. Bennet, impossible, when I am not acquainted with him myself; how can you be so teasing?”	Mrs. Bennet
	“I honour your circumspection. [...] I will take it on myself.”	Mr. Bennet
	The girls stared at their father. Mrs. Bennet said only, “Nonsense, nonsense!”	Mrs. Bennet

Table 1: Quotes where speakers are mentioned explicitly, by anaphor, or implicitly (conversationally).

2 Related Work

Early work in quote attribution focused on identifying spans associated with content (quotes), sources (mentions), and cues (speech verbs) in newswire data. This is the approach taken by Pareti et al. (2012; 2013). More recent work by Almeida et al. (2014) performed entity-level quote attribution and showed that a joint model of coreference and quote attribution can help both tasks.

In the literary domain, Glass and Bangay (2007) did early work modeling both the mention-level and entity-level tasks using a rule-based system. However, their system relied on identifying a main speech verb to then identify the actor (i.e. the mention) and link to the speaker (i.e. the entity) from a character list. This system worked very well but was limited to explicitly cued speakers and did not address implicit speakers at all.

Elson and McKeown (2010) took important first steps towards automatic quote attribution. They formulated the task as one of *mention* identification in which the goal was to link a quote to the mention of its speaker. Their method achieved 83.0% accuracy overall, but used gold-label information at test time. Their corpus, the Columbia Quoted Speech Corpus (CQSC), is the most well-known corpus and was used by follow-up work. However, a result of their Mechanical Turk-based labeling strategy was that this corpus contains many unannotated quotes (see Table 4).

O’Keefe et al. (2012) also treated quote attribution as mention identification, using a sequence labeling approach. Their approach was successful in the news domain but it failed to beat their baseline in the literary domain (53.5% vs 49.8%

Quote Types	<i>Emma</i>	<i>The Steppe</i>
with mention	546 (74.4%)	371 (59.6%)
with speaker	491 (66.9%)	258 (41.5%)

Table 4: Coverage of the CQSC labels

accuracy). This work quantitatively showed that quote attribution in literature was fundamentally different from the task in newswire.

We compare against He et al. (2013), the previous state-of-the-art system for quote attribution. They re-formulated quote attribution as quote-speaker labeling rather than quote-mention labeling. They used a supervised learner and a generative actor topic model (Celikyilmaz et al., 2010) to achieve accuracies ranging from 82.5% on *Pride & Prejudice* to 74.8% on *Emma*.

3 Data: The QuoteLi3 Corpus

We build upon the datasets of He et al. (2013) and Elson and McKeown (2010) to create a comprehensive new dataset of quoted speech in literature: QuoteLi3. This dataset covers 3 novels and 3103 individual quotes, each linked to speaker and mention for a total of 6206 labels, more than 3000 of which are newly annotated. It is composed of expert-annotated dialogue from Jane Austen’s *Pride and Prejudice*, *Emma*, and Anton Chekhov’s *The Steppe*.

3.1 Previous Datasets

The datasets described in section 2 are valuable but incomplete and hard to integrate with one another given their different designs.

The Columbia Quoted Speech Corpus is a large dataset that includes both quote-mention and

Novel	He et al.		CQSC (Elson and McKeown, 2010)		QuoteLi3	
	q-mention	q-speaker	q-mention	q-speaker	q-mention	q-speaker
<i>Pride and Prejudice</i>	○	●			●	●
<i>Emma</i>			●	●	●	●
<i>The Steppe</i>			●	●	●	●

Table 2: Label coverage per novel: ● is full, ● is partial, and ○ is no coverage of annotations.

Quote Type	QuoteLi3 (uncollapsed)			QuoteLi3 (collapsed)			He et al.		
	<i>P & P</i>	<i>Emma</i>	<i>The Steppe</i>	<i>P & P</i>	<i>Emma</i>	<i>The Steppe</i>	<i>P & P</i>	<i>Emma</i>	<i>The Steppe</i>
Explicit (ES)	555	240	278	326	128	184	305	106	112
Anaphoric (AS)	528	132	180	309	73	106	292	55	39
pronoun (AS(p))	405	112	106	241	58	58			
other (AS(o))	123	20	74	68	15	48			
Implicit (IS)	664	362	164	655	357	158	663	236	93
Total	1747	734	622	1290	558	448	1260	397	244
<i>All</i>		3103			2296			1901	

Table 3: Breakdown of our dataset by novel and type of quote (uncollapsed). For comparison with the dataset from He et al. (2013), we provide the collapsed statistics assuming one speaker per paragraph.

quote-speaker labels (Elson and McKeown, 2010). It suffers from problems often associated with crowdsourced labels and the use of low-accuracy tools. In this corpus, quote-mention labels were gathered from Mechanical Turk, where each quote was linked to a mention by 3 different annotators. Elson and McKeown (2010) report that 65% of the quotes in CQSC had unanimous agreement and that 17.6% of the quotes in this corpus were unlabeled. To generate quote-speaker labels, an off-the-shelf coreference tool³ was used to link mentions and form coreference chains. We find that 57.8% of the quotes in this corpus either i) have no speaker label (48.1%) or ii) the speaker cannot be linked to a known character entity (9.7%). O’Keefe et al. (2012) find that 8% of quotes with speaker labels are incorrectly labeled. Our analysis of the relevant part of CQSC for this work is shown in Table 4.

The data from He et al. (2013) includes high-quality speaker labels but lacks quote-mention labels. There is no overlap in the data provided by He et al. (2013) and CQSC, but this work did evaluate their system on a subset of CQSC. This dataset assumes that all quoted text within a paragraph should be attributed to the same speaker.⁴ While this assumption is correct for *Pride and Prejudice*, it is incorrect for novels like *The Steppe*, which use more complex conversa-

tional structures⁵. This assumption leads to a problematic method of system evaluation in which all quotes within a paragraph are considered in the gold labels to be one quote, even if they were in fact uttered by different characters. We refer to this strategy as having “collapsed” quotes in our evaluations and present it for the purpose of providing a faithful comparison to previous work.

In QuoteLi3 we add the annotations that are missing from both datasets and correct the existing ones where necessary. A summary of the annotations included in this dataset and comparison to the previous data that we draw from is described in Table 2. Our final dataset is described in Table 3. It features a complete set of annotations for both quote-mention and quote-speaker labels.

3.2 Annotation

Two of the authors of the paper were the annotators of our dataset. They used annotation guidelines consisting of an example excerpt and a description, which is included in the supplementary materials §A.5. The annotators were instructed to identify the speaker (from a character list) for each quote and to identify the mention that most directly helped them determine the speaker. Unlike Elson and McKeown (2010), mentions can be pronouns and vocatives, not just explicit name referents. Mentions that were closer to the quote and speech verbs were favored over indirect mentions (such as those in conversational chains). Figure 1 shows an example from *Pride and Prejudice*.

Annotation was done using a browser-based an-

³Even current state-of-the-art coreference tools achieve just over 65% average F1 scores (Clark and Manning, 2016).

⁴For first-level quotes, there is typically just one speaker per paragraph. This assumption breaks down in some cases and it is very rarely true for nested quotes.

⁵See supplemental section A.1.

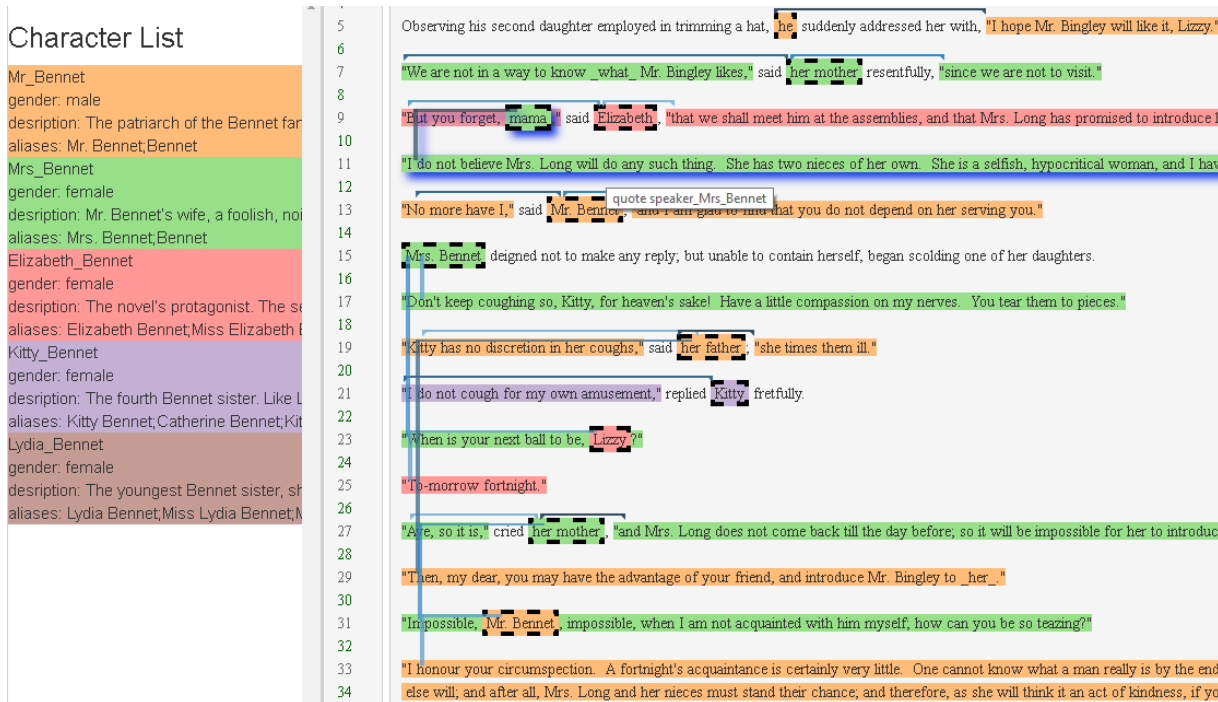


Figure 1: Conversation from *Pride and Prejudice* annotated with our annotation tool. Speakers are indicated by color, mentions are marked by dashed outlines, and quote-to-mention links by blue lines.

notation tool developed by the authors. Previously developed tools were either not designed for the task (BRAT (Stenetorp et al., 2012), WebAnno (Yimam et al., 2013), CHARLES (Vala et al., 2016)) or unavailable (He et al., 2013). One problem with the CQSC annotations was that the annotators were shown short snippets that lacked the context to determine the speaker and no character list. We designed our tool to provide context and a character list including name, aliases, gender, and description of the character. Similar to CHARLES, the character list is not static and the annotator can add to the list of characters. Our tool also features automatic data consistency checks such as ensuring that all quotes are linked to a mention.

Our expert annotators achieved high inter-annotator agreement with a Cohen’s κ of .97 for quote-speaker labels and a κ of .95 for quote-mention labels.⁶ To preserve the QuoteLi3 data for train, dev, and testing sets, we calculated this inter-annotator agreement on excerpts from *Alice in Wonderland* and *The Adventures of Huckleberry Finn* containing 176 quotes spoken by 10 characters, chosen to be similar to the data found in QuoteLi3.

⁶The reported agreement is the average of the Cohens kappas from these passages.

3.3 Statistics

Table 3 shows the statistics of our annotated corpus. Unlike He et al. (2013), we do not assume that all quotes in the same paragraph are spoken by the same speaker. To compare with the dataset used by He et al. (2013), we provide the collapsed statistics as well. As Table 3 shows, we have roughly the same number of annotated quotes for *Pride and Prejudice* as He et al. (2013). For *Emma* and *The Steppe*, which were taken from the CQSC corpus, we have considerably more quotes because of our added annotations (see Table 4).

4 The Quote Attribution Task

The task of quote attribution can be summarized as “who said that?” Given a text as input, the final output is a speaker for each uttered quote in the text. We assume that all quotes have been previously identified. O’Keefe et al. (2012) find that regular-expression approaches to quote detection yield over 99% accuracy for clean English-language data. A number of other approaches to quote detection have been studied in recent years for more complex data (Pouliquen et al., 2007; Pareti et al., 2013; Muzny et al., 2016; Scheible et al., 2016). Following He et al. (2013), we assume that there is a predefined list of characters available, with the name, aliases, and gender of each

character.⁷

Some key challenges in quote attribution are resolving anaphora (i.e., coreference) and following conversational threads. Literature often follows specific patterns that make some quotes easier to attribute than others. Therefore, an approach that anchors conversations on easily identifiable quotes can outperform approaches that do not.

Figure 1 shows an example of a complex conversation at the beginning of *Pride and Prejudice*. This example illustrates the spectrum of easy to difficult cases found in the task: simple explicit named mention (lines 9, 13, 21), nominal mentions (lines 7, 19, 27), and pronoun mentions (line 5). Sometimes explicitly named mentions embedded in more complex sentences can still be challenging as they require good dependency parses. This example also illustrates a conversational chain with alternating speakers between Mrs. Bennet and Elizabeth Bennet (lines 7 to 11), and between Mr. Bennet and Mrs. Bennet (lines 27 to 34). In this case, vocatives (expressions that indicate the party being addressed) are cues for who the other speaker is (lines 9, 23, 31). When the simple alternation pattern is broken, explicit speech verbs with the speaking character are specified. To summarize, there are several explicit cues and some easy cases in a conversation that can be leveraged to make the hard cases easier to address.

First, consider the quote→mention linking subtask. This is an inherently ambiguous task (i.e. any mention from the same coreference chain is valid,) but we know that if the target quote is linked to the annotated mention that this is one correct option. This means that the evaluation of the quote→mention stage is a lower-bound. In other words, since a given quote may have multiple mentions that could be considered correct, our system may choose a “wrong” mention for a quote but link it to the correct speaker in the end. Thus, if our mention→speaker system could perfectly resolve every mention to its correct speaker, our overall quote attribution system would be guaranteed to get at minimum the same results as the quote→mention stage.

The quote→speaker task can be tackled directly without addressing quote→mention, but identifying a mention associated with the speaker allows us to incorporate key outside information. An

⁷Character lists are available on sites like sparknotes.com. The automatic extraction of characters from a novel has been identified as a separate problem (Vala et al., 2015).

other advantage of this approach is that we can then separately analyze and improve the performance of the two stages.

Therefore we evaluate both subtasks to give a more complete picture of when the system fails and succeeds. We use precision, recall, and F1 so that we can tune the system for different needs.

5 Approach

Our model is a two-stage deterministic pipeline. The first stage links quotes to specific mentions in the text and the second stage matches mentions to the entity that they refer to.

By doing both quote→mention and mention→entity linking, our system is able to leverage additional contextual information, resulting in a richer, labeled output. Its modular design means that it can be easily updated to account for improvements in various sub-areas such as coreference resolution. We use a sieve-based architecture because having accurate labels for the easy cases allows us to first find anchors that help resolve harder, often conversational, cases. Sieve-based systems have been shown to work well for tasks like coreference resolution (Raghunathan et al., 2010; Lee et al., 2013), entity linking (Hajishirzi et al., 2013), and event temporal ordering (Chambers et al., 2014).

5.1 Quote→Mention

The quote→mention stage is a series of deterministic sieves. We describe each in detail in the following sections and show examples in Table 5.

Trigram Matching This sieve is similar to patterns used in Elson and McKeown (2010). It uses patterns like Quote-Mention-Verb (e.g. ``...'' she said) where the mention is either a character name or pronoun to isolate the mention. Other patterns include Quote-Verb-Mention, Mention-Verb-Quote, and Verb-Mention-Quote.

Dependency Parses The next sieve in our pipeline inspects the dependency parses of the sentences surrounding the target quote. We use the enhanced dependency parses (Schuster and Manning, 2016) produced by Stanford CoreNLP (Chen and Manning, 2014) to extract all verbs and their dependent `nsubj` nodes. If the verb is a common speech verb⁸ and its `nsubj` relation points to a

⁸This list of verbs as well as the family relation nouns list are available in supplemental section A.4.

Sieve	Example
Trigram Matching	“They have none of them much to recommend them,” replied he .
Dependency Parses	Mrs. Bennet said only, “Nonsense, nonsense!”
Single Mention Detection	... Elizabeth impatiently. “There has been many a one, I fancy, overcome in the same way. I wonder who first discovered the efficacy of poetry in driving away love!”
Vocative Detection	“My dear Mr. Bennet ,...” “Is that his design in settling here?”
Paragraph Final Mention Linking	After a silence of several minutes, he came towards her in an agitated manner, and thus began, “In vain have I struggled...”
Supervised Sieve	–
Conversation Detection	“Aye, so it is,” cried her mother ... “Then, my dear, you may have the advantage of your friend, and introduce Mr. Bingley to her.” “Impossible, Mr. Bennet, impossible, when I am not acquainted with him myself; how can you be so teasing?”
Loose Conversation Detection	“I will not trust myself on the subject,” replied Wickham ; “I can hardly be just to him.” Elizabeth was again deep in thought, and after a time exclaimed, “To treat in ... the favourite of his father!” She could have added, “A young man, too,... being amiable”– but she contented herself with, “and one, too, ... in the closest manner!” “We were born in the same parish, within the same park; the greatest part of our youth was passed together;...”

Table 5: Quote→Mention sieves and example quotes that they apply to.

Sieve	Example
Exact Name Match	“ <i>Do you really think so?</i> ” cried Elizabeth , brightening up for a moment.
Coreference Disambiguation	“ <i>You are uniformly charming!</i> ” cried he , with an air of awkward gallantry;
Conversational Pattern	“ <i>Impossible, Mr. Bennet, impossible ...</i> ” (Mrs. Bennet) “I honour your circumspection...I will take it on myself.” (Mr. Bennet) The girls stared at their father. Mrs. Bennet said only, “Nonsense, nonsense!” (Mrs. Bennet)
Family Noun Vocative Disambiguation	“...You know, sister , we agreed long ago never to mention a word about it. And so, is it quite certain he is coming?” “ <i>You may depend on it,</i> ” replied the other ...
Majority Speaker	–

Table 6: Mention→Speaker Sieves and example quotes that they apply to. Bold text indicates where the speaker information comes from while italic text indicates the target quote being labeled.

character name, a pronoun, or an animate noun,⁹ we assign the quote to the target mention.

Single Mention Detection If there is only a single mention in the non-quote text in the paragraph of the target quote, link the quote to that mention.

Vocative Detection If the preceding quote contains a vocative pattern (see supplemental section A.2), link the target quote to that mention. Vocative detection only matches character names and animate nouns.

Paragraph Final Mention Linking If the target quote occurs at the end of a paragraph, link it to the final mention occurring in the preceding sentence.

Conversational Pattern If a quote in paragraph n has been linked to mention m_n , then this sieve links an unattributed quote two paragraphs ahead, $n + 2$, to mention m_n if they appear to be in conversation. We consider two quotes “in conversation” if the paragraph between is also a quote, and

⁹The list of animate nouns is from Ji and Lin (2009).

the quote in paragraph $n + 2$ appears without additional (non-quote) text.

Loose Conversational Pattern We include a looser form of the previous sieve as a final, high-recall, step. If a quote in paragraph n has been linked to mention m_n , then this sieve links quotes in paragraph $n + 2$ to m_n without restriction.

5.2 Mention→Speaker

The second stage of our system involves linking the mentions identified in the first stage to a speaker entity. We again use several simple, deterministic sieves to determine the entity that each mention and quote should be linked to. A description of these sieves and example mentions and quotes that they are applied to appears in Table 6.

For the following sieves, we construct an ordered list of *top_speakers* by counting proper name and pronoun mentions around the target quote. If gender for the target quote’s speaker can be determined either by the gender of a pronoun mention or the gender of an animate noun (Bergsma and

Lin, 2006), this information is used to filter the candidate speakers in the *top_speakers* list.

We use a window size from 2000 tokens before the target quote to 500 tokens after the target quote. If no speakers matching in gender can be found in this window, it is expanded by 2000 tokens on both sides.

Exact Name Match If the mention that a quote is linked to matches a character name or alias in our character list, label the quote with that speaker.

Coreference Disambiguation If the mention is a pronoun, we attempt to disambiguate it to a specific character using the coreference labels provided by BookNLP (Bamman et al., 2014).

Conversational Pattern Similarly as in the quote→mention section, we match a target quote to the same speaker as a quote in paragraph $n + 2$, if they are in the same conversation and it is labeled. Next, we match it to the quote in paragraph $n - 2$ if they are in the same conversation and it is labeled. This sieve receives gender information from the mention that the target quote is linked to.

Family Noun Vocative Disambiguation If the target quote is linked to a vocative in the list of family relations (e.g. “papa”), pick the first speaker in *top_speakers* that matches the last name of the speaker of the quote containing the vocative.

Majority Speaker If none of the previous sieves identified a speaker for the quote, label the quote with the first speaker in the *top_speakers* list.

6 Experiments

In all experiments, we divide the data as follows: *Pride and Prejudice* is split as in He et al. (2013) with chapters 19-26 as the test set, 27-33 as the development set, and all others as training. *Emma* and *The Steppe* are not used for training.

6.1 Baseline

As a baseline, for the quote→mention stage we choose the mention that is closest to the quote in terms of token distance. This is similar to the approach taken in BookNLP (Bamman et al., 2014), in which quotes are attributed to a mention by first looking for the closest mention in the same sentence to the left and right of the quote, then before a hard stop or another quote to the left and right of the target quote. For the mention→speaker stage,

Test	ES	AS(p)	AS(o)	IS	All
<i>P & P</i>	98.4	77.3	42.9	82.3	85.1
<i>Emma</i>	92.1	62.5	35.0	71.5	75.9
<i>The Steppe</i>	97.5	67.0	14.9	60.4	72.7

Table 9: Breakdown of the accuracy of our system per type of quote (see Table 3) in each test set.

we use the Exact Name Match and Coreference Disambiguation sieves.

6.2 Comparison to Previous Work

Table 7 shows a direct comparison of our work versus the previous systems. We replicate the test conditions used by He et al. (2013) as closely as possible in this comparison.

In this comparison, the evaluations based on CQSC are of non-contiguous subsets of the quotes that are also not necessarily the same between our work and the previous work. As discussed in section 3, CQSC provides an incomplete set of quote-speaker labels. In this work we follow the same methodology as He et al. (2013) to extract a test set of unambiguously labeled quotes by using a list of character names to identify those that are unambiguously labeled. In section 7, we analyze *The Steppe* and *Emma* more thoroughly, showing that this method results in an easier subset of the quotes in these novels.

Our preferred evaluation, shown in Table 8, differs from previous evaluations in four important ways. We hope that this work can establish consistent guidelines for attributing quotes and evaluating system performance to encourage future work.

- Each quote is attributed separately.¹⁰
- The test sets are composed of every quote from the test portion of each novel, no subsets are used because of incomplete annotations.¹¹
- No gold data is used at test time.¹²
- Precision and recall are reported in preference to accuracy for a more fine-grained understanding of the underlying system.

¹⁰This is in contrast to the work of He et al. (2013)

¹¹This is in contrast to the work of Elson and McKeown (2010) and He et al. (2013). The work of O’Keefe et al. (2012) is the only previous work to augment the unlabeled portions of CQSC. They achieved 53.3% accuracy on CQSC from a rule-based system similar to our baseline. This data is not available.

¹²Gold data was used at test time by Elson and McKeown (2010) who achieved 83.0% accuracy on the CQSC.

Test	He et al.	Baseline	This work	+ supervised
<i>Pride and Prejudice</i>	82.5	45.3	83.6	85.2
<i>Emma</i>	74.8*	40.7*	75.3*	76.1*
<i>The Steppe</i>	80.3*	66.7*	81.8*	83.8*

Table 7: Comparison with previous work. This table reports accuracy and comes with some caveats: * indicates that a non-contiguous subset of the quotations were used (not all subsets are guaranteed to be the same as described in section 6.2), and all quotes within the same paragraph were collapsed. *Emma* and *The Steppe* come from CQSC. All systems are trained on *Pride and Prejudice*.

System	Test	Quote→Mention			Mention→Speaker			Accuracy
		P	R	F1	P	R	F1	
+supervised	<i>Pride and Prejudice</i>	86.7	93.5	89.9	85.1	100	92.0	85.1
+supervised	<i>Emma</i>	75.2	85.2	79.9	75.9	100	86.3	75.9
+supervised	<i>The Steppe</i>	81.7	88.6	85.0	72.7	100	84.2	72.7
	Average	81.2	89.1	84.9	77.9	100	87.5	
+precision	<i>Pride and Prejudice</i>	90.2	80.1	84.9	92.1	70.9	80.1	
+precision	<i>Emma</i>	84.6	68.3	75.6	85.7	59.0	69.9	
+precision	<i>The Steppe</i>	92.5	75.3	83.0	93.3	65.5	77.0	
	Average	89.1	74.6	81.2	90.4	65.1	75.7	

Table 8: Precision, recall, and F-Score of our systems on un-collapsed quotations and the fully annotated test sets from the QuoteLi3 dataset.

6.3 Adding a Supervised Component

To test how orthogonal our two-stage approach is to previous systems, we experiment by adding a supervised sieve to the quote→mention stage. We train a binary classifier, using a maxent model to distinguish between the correct and incorrect candidate mentions.

Candidate Mentions We take as candidate mentions all token spans corresponding to names, pronouns, and animate nouns in a one-paragraph range on either side of the quote. Names are determined by scanning for matches to the character list. We restrict pronouns to singular gendered pronouns, i.e. ‘he’ or ‘she’.

Features We featurize each (quote, mention) pair based on attributes of the quote, mention, and how far apart they are from one another. These features largely align with previous work and can be found in supplemental section A.3 (Elson and McKeown, 2010; He et al., 2013).

Prediction At test time our model predicts for each quote whether each candidate mention is or is not the correct mention to pair with that quote. If the model predicts more than one mention to be correct, we take the most confident result.

This sieve goes just before the conversation pat-

tern detection sieves in the quote→mention stage (see Table 5). This forms our +*supervised* system.

6.4 Creating a High-Precision System

One advantage of our sieve design is that we can easily add and remove sieves from our pipeline. This means that we can determine the combination of sieves that result in the system that achieves the highest precision with respect to the final speaker label. We use an ablation test to find the combination of sieves with the highest precision (95.6%) for speaker labels on the development set from *Pride and Prejudice*. These results are achieved by removing the Loose Conversation Detection sieve for the quote→mention stage and keeping only the Exact Name Match and Coreference Disambiguation sieves for the mention→speaker stage.

Together, these sieves create a system that we call +*precision* that emphasizes overall precision rather than F-score or accuracy.

7 Results

We show that a simple deterministic system can achieve state-of-the-art results. Adding a lightweight supervised component improves the system across all test sets. The sieve design allows us to create a high precision system that might be more appropriate for real-world applications that

value precision over recall.

The results in Table 8 confirm that the subset of test quotes from *Emma* and *The Steppe* used in previous work were an easier subset of the whole set of quotations. When evaluating based off of the whole set of quotations, we lose 0.2 and 11.1 points of accuracy for *Emma* and *The Steppe*, respectively. As we show in Table 4, *The Steppe* is missing a significant portion (50.9%) of the annotations whereas *Emma* is missing 28.6%. Our error analysis shows us that *The Steppe* features more complicated conversation patterns than the novels of Jane Austen, which makes the task of quote attribution more difficult.

One type of error analysis we performed was inspecting the accuracy of our system by quote type. As seen in Table 9, the main challenge lies in identifying anaphoric and implicit speakers. We find that resolving non-pronoun anaphora is much more challenging for our system than pronouns. This is because the only mechanism for dealing with these mentions is the Family Noun Vocative Disambiguation sieve; otherwise, the only information we gather from them is gender information. This indicates that adding information about the social network of a novel and attributes of each character (such as job and relationships to other characters) would further increase system performance.

8 Conclusion

In this paper, we provided an improved, consistently annotated dataset for quote attribution with both quote-mention and quote-speaker annotations. We described a two-stage quote attribution system that first links quotes to mentions and then mentions to speakers, and showed that it outperforms the existing state-of-the-art. We established a thorough evaluation and showed how our system can be tweaked for higher precision or refined with a supervised sieve for better overall performance.

A clear direction for future work is to expand the dataset to a more diverse set of novels by leveraging our annotation tool on Mechanical Turk or other crowdsourcing platforms. This work has also provided the background to see the pitfalls that a dataset produced in such a way might encounter. For example, annotators could label mentions and speakers separately, and examples with high uncertainty could be transferred to expert annotators. An expanded dataset would allow us to

evaluate how well our system generalizes to other novels and also allow us to train better models. Another interesting direction is to eliminate the use of predefined character lists by automatically extracting the list of characters (Vala et al., 2015).

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-114747, and by the NSF via IIS IIS-1514268. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors thank their anonymous reviewers and the Stanford NLP group for their helpful feedback.

References

- Mariana S.C. Almeida, Miguel B. Almeida, and André FT Martins. 2014. A joint model for quotation attribution and coreference resolution. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.
- David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Asli Celikyilmaz, Dilek Hakkani-Tur, Hua He, Greg Kondrak, and Denilson Barbosa. 2010. The actor-topic model for extracting social networks in literary narrative. In *Proceedings of NIPS Workshop: Machine Learning for Social Computing*.
- Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.
- Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*.
- Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of Association for Computational Linguistics (ACL)*.

- David K. Elson and Kathleen McKeown. 2010. Automatic attribution of quoted speech in literary narrative. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*.
- Kevin Glass and Shaun Bangay. 2007. A naïve, salience-based method for speaker identification in fiction books. In *Proceedings of the 18th International Symposium of the Pattern Recognition Association of South Africa (PRASA)*.
- Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke S. Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceedings of EMNLP*, pages 289–299.
- Hua He, Denilson Barbosa, and Grzegorz Kondrak. 2013. Identification of speakers in novels. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Heng Ji and Dekang Lin. 2009. Gender and animacy knowledge discovery from web-scale n-grams for unsupervised person mention detection. In *Proceedings of Pacific Asia Conference on Language, Information and Computation (PACLIC)*.
- Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.
- Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Grace Muzny, Mark Algee-Hewitt, and Dan Jurafsky. 2016. The dialogic turn and the performance of gender: the English canon 1782-2011. In *Proceedings of Digital Humanities*, pages 296–299.
- Tim O’Keefe, Silvia Pareti, James R. Curran, Irena Koprinska, and Matthew Honnibal. 2012. A sequence labelling approach to quote attribution. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*.
- Silvia Pareti, Timothy O’Keefe, Ioannis Konstas, James R. Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*.
- Silvia Pareti. 2012. A database of attribution relations. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*.
- Bruno Pouliquen, Ralf Steinberger, and Clive Best. 2007. Automatic detection of quotations in multilingual news. In *Proceedings of Recent Advances in Natural Language Processing*.
- Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*.
- Christian Scheible, Roman Klinger, and Sebastian Padó. 2016. Model architectures for quotation detection. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.
- Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. Bennet, his coachman, and the archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*.
- Hardik Vala, Stefan Dimitrov, David Jurgens, Andrew Piper, and Derek Ruths. 2016. Annotating characters in literary corpora: A scheme, the CHARLES tool, and an annotated novel. In *Proceedings of the 10th edition of the Language Resources and Evaluation Conference (LREC)*.
- Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.
- Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of Association for Computational Linguistics (ACL) System Demonstrations*.

A Supplemental Material

A.1 Nested Conversation Example



Figure 2: An example paragraph that contains multiple speakers from *The Steppe*

Figure 2 shows a screen shot of our annotation tool displaying a paragraph with a complex conversational structure from *The Steppe*.

A.2 Vocative Patterns

Pattern	Example
between , and !	, Nastasya!
between , and ?	, Mr. Bennet?
between , and .	, Yegorushka.
between , and ;	, papa;
between , and ,	, Emma,
between “ and ,	“Father Christopher,
between , and ”	, mother”
after the word “dear”	Dear Lydia
between “oh” and !	Oh Henry!

Table 10: Vocative patterns for extracting mentions.

A.3 Supervised Classifier Features

We used the following features in our supervised classifier:

- *Distance*: token distance, ranked distance (relative to mentions), paragraph distance (left paragraph and right paragraph separate)
- *Mention*: Number of quotes in the mention paragraph, number of words in mention paragraph, the order of the mention within the paragraph (compared to other mentions), whether the mention is within conversation

(i.e. no non-quote text in the same paragraph), whether the mention is within a quote, POS of the previous and next words.

- *Quote*: the length of the quote, the order of the quote (i.e. whether it is the first or second quote in a paragraph), the number of words in the paragraph, number of names in the paragraph, whether the quote contains text in it, whether the present quote contains the name of the mention (if mention is a name).

A.4 Words Lists

Common Speech Verbs Similar to He et al. (2013), we use *say*, *cry*, *reply*, *add*, *think*, *observe*, *call*, and *answer*, present in the training data from *Pride and Prejudice*.

Family Relation Nouns ancestor aunt bride bridegroom brother brother-in-law child children dad daddy daughter daughter-in-law father father-in-law fiancée grampa gramps grandchild grandchildren granddaughter grandfather grandma grandmother grandpa grandparent grandson granny great-granddaughter great-grandfather great-grandmother great-grandparent great-grandson great-aunt great-uncle groom half-brother half-sister heir heiress husband mama mom mommy mother mother-in-law nana nephew niece pa papa parent pop second cousin sister sister-in-law son son-in-law stepbrother stepchild stepchildren stepdad stepdaughter stepfather stepmom stepmother stepsister stepson uncle wife

A.5 Annotation Guidelines

- Each quote should be annotated with the character that is that quote’s speaker.
- Each quote should be linked to a mention that is the most obvious indication of that quote’s speaker.
 - Quotes can be linked to mentions inside other quotes.
 - Multiple quotes may be linked to the same mention.
- Mentions should also be annotated with the character that they refer to.
 - If a character’s name is meaningfully associated with an article (e.g. “...,” said *the Bear*), include that article in the mention.

Out-of-domain FrameNet Semantic Role Labeling

Silvana Hartmann^{§†}, Ilia Kuznetsov[†], Teresa Martin^{§†}, Iryna Gurevych^{§†}

§Research Training Group AIPHES

†Ubiquitous Knowledge Processing (UKP) Lab

Department of Computer Science, Technische Universität Darmstadt

<http://www.ukp.tu-darmstadt.de>

Abstract

Domain dependence of NLP systems is one of the major obstacles to their application in large-scale text analysis, also restricting the applicability of FrameNet semantic role labeling (SRL) systems. Yet, current FrameNet SRL systems are still only evaluated on a single in-domain test set. For the first time, we study the domain dependence of FrameNet SRL on a wide range of benchmark sets. We create a novel test set for FrameNet SRL based on user-generated web text and find that the major bottleneck for out-of-domain FrameNet SRL is the frame identification step. To address this problem, we develop a simple, yet efficient system based on distributed word representations. Our system closely approaches the state-of-the-art in-domain while outperforming the best available frame identification system out-of-domain. We publish our system and test data for research purposes.¹

1 Introduction

Domain dependence is a major problem for supervised NLP tasks such as FrameNet semantic role labeling (SRL): systems generally exhibit a strong performance drop when applied to test data from a different distribution than the training data. This prohibits their large-scale use in language technology applications.

The same problems are expected for FrameNet SRL, but due to a lack of datasets, state-of-the-art FrameNet SRL is only evaluated on a single in-domain test set, see e.g. Das et al. (2014) and FitzGerald et al. (2015).

In this work, we present the first comprehensive study of the domain dependence of FrameNet SRL

on a range of benchmark datasets. This is crucial as the demand for semantic textual analysis of large-scale web data keeps growing.

Based on FrameNet (Fillmore et al., 2003), FrameNet SRL extracts frame-semantic structures on the sentence level that describe a specific situation centered around a semantic predicate, often a verb, and its participants, typically syntactic arguments or adjuncts of the predicate. The predicate is assigned a *frame* label, essentially a word sense label, that defines the situation and determines the *semantic roles* of the participants. The following sentence from FrameNet provides an example of the *Grinding* frame and its roles:

[The mill]_{Grinding_cause} **grinds**_{Grinding} [the malt]_{Patient} [to grist]_{Result}.

FrameNet SRL consists of two steps, frame identification (frameId), assigning a frame to the current predicate, and role labeling (roleId), identifying the participants and assigning them role labels licensed by the frame. The frameId step reduces the hundreds of role labels in FrameNet to a manageable set of up to 30 roles. Thus, FrameNet SRL differs from PropBank SRL (Carreras and Màrquez, 2005), that only uses a small set of 26 syntactically motivated role labels and puts less weight on the predicate sense. The advantage of FrameNet SRL is that it results in a more fine-grained and rich interpretation of the input sentences which is crucial for many applications, e.g. reasoning in online debates (Berant et al., 2014).

Domain dependence is a well-studied topic for PropBank SRL. However, to the best of our knowledge, there exists no analysis of the performance of modern FrameNet SRL systems when applied to data from new domains.

In this work, we address this problem as follows: we introduce a new benchmark dataset YAGS

¹www.ukp.tu-darmstadt.de/ood-fn-srl

(Yahoo! Answers Gold Standard), which is based on user-generated questions and answers and exemplifies an out-of-domain application use case. We use YAGS, along with other out-of-domain test sets, to perform a detailed analysis of the domain dependence of FrameNet SRL using *Semafor* (Das et al., 2014; Kshirsagar et al., 2015) to identify which of the stages of FrameNet SRL, *frameId* or *roleId*, is particularly sensitive to domain shifts. Our results confirm that the major bottleneck in FrameNet SRL is the frame identification step. Motivated by that, we develop a simple, yet efficient frame identification method based on distributed word representations that promise better domain generalization. Our system’s performance matches the state-of-the-art in-domain (Hermann et al., 2014), despite using a simpler model, and improves on the out-of-domain performance of *Semafor*.

The contributions of the present work are two-fold: 1) we perform the first comprehensive study of the domain generalization capabilities of open-source FrameNet SRL, and 2) we propose a new frame identification method based on distributed word representations that enhances out-of-domain performance of frame identification. To enable our study, we created YAGS, a new, substantially-sized benchmark dataset for the out-of-domain testing of FrameNet SRL; we publish the annotations for the YAGS benchmark set and our frame identification system for research purposes.

2 Related work

The domain dependence of FrameNet SRL systems has been only studied sparsely, however, there exists a large body of work on out-of-domain PropBank SRL, as well as on general domain adaptation methods for NLP. This section briefly introduces some of the relevant approaches in these areas, and then summarizes the state-of-the-art in FrameNet frame identification.

Domain adaptation in NLP Low out-of-domain performance is a problem common to many supervised machine learning tasks. The goal of domain adaptation is to improve model performance on the test data originating from a different distribution than the training data (Søgaard, 2013). For NLP, domain adaptation has been studied for various tasks such as POS-tagging and syntactic parsing (Daumé III, 2007; Blitzer et al., 2006). For the complex task of SRL, it is strongly associated with PropBank, because

the corresponding CoNLL shared tasks promote out-of-domain evaluation (Surdeanu et al., 2008; Hajič et al., 2009). In the shared tasks, in-domain newspaper text from the WSJ Corpus is contrasted to out-of-domain data from fiction texts in the Brown Corpus. Most of the participants in the shared tasks do not consider domain adaptation and report systematically lower scores for the out-of-domain data (Hajič et al., 2009).

Representation learning has been successfully used to improve on the CoNLL shared task results (Huang and Yates, 2010; FitzGerald et al., 2015; Yang et al., 2015). Yang et al. (2015) report the smallest performance difference (5.5 points in F_1) between in-domain and out-of-domain test data, leading to the best results to date on the CoNLL 2009 out-of-domain test. Their system learns common representations for in-domain and out-of-domain data based on deep belief networks.

Domain dependence of FrameNet SRL The FrameNet 1.5 fulltext corpus, used as a standard dataset for training and evaluating FrameNet SRL systems, contains texts from several domains (Ruppenhofer et al., 2010). However, the standard data split used to evaluate modern systems (Das and Smith, 2011) ensures the presence of all domains in the training as well as test data and cannot be used to assess the systems’ ability to generalize. Moreover, all the texts in the FrameNet fulltext corpus, based on newspaper and literary texts, are post-edited and linguistically well-formed. The FrameNet test setup thus cannot provide information on SRL performance on less edited out-of-domain data, e.g. user-generated web data.

There are few studies related to the out-of-domain generalization of FrameNet SRL. Johansson and Nugues (2008) evaluate the impact of different parsers on FrameNet SRL using the Nuclear Threats Initiative (NTI) data as an out-of-domain test set. They observe low domain generalization abilities of their supervised system, but find that using dependency parsers instead of constituency parsers is beneficial in the out-of-domain scenario. Croce et al. (2010) use a similar in-domain/out-of-domain split to evaluate their approach to open-domain FrameNet SRL. They integrate a distributional model into their SRL system to generalize lexicalized features to previously unseen arguments and thus create an SRL system with a smaller performance gap between in-domain and out-of-domain test data (only 4.5 percentage points F_1).

Note that they only evaluate the role labeling step. It is not transparent how their results would transfer to the current state-of-the-art SRL systems that already integrate methods to improve generalization, for instance using distributed representations.

Palmer and Sporleder (2010) analyze the FrameNet 1.3 training data coverage and the performance of the Shalmaneser SRL system (Erk and Padó, 2006) for frame identification on several test sets across domains, i.e. the PropBank and NTI parts of the FrameNet fulltext corpus and the fictional texts from the SemEval-2007 shared task (Baker et al., 2007). Having observed that the majority of errors results from coverage gaps in FrameNet, they suggest to focus on developing frame identification systems that generalize well to new domains. Our observations support their findings and show that the problem still persists even when modern SRL methods and the extended FrameNet 1.5 lexicon are used.

Søgaard et al. (2015) annotate 236 tweets with FrameNet labels to apply SRL to knowledge extraction from Twitter. They report that the frameId performance of `Semafor 2.1` (Das et al., 2010) on the new test set is similar to its performance on the SemEval-2007 newswire test set (Baker et al., 2007). For full SRL, there are large differences: F_1 reaches only 25.96% on the Twitter set compared to the 46.5% reported by Das et al. (2010) on the in-domain set. These results show that there is ample room for improvement for SRL on Twitter data.

Recent FrameNet SRL systems are not evaluated in the context of their domain dependence: Kshirsagar et al. (2015) use the domain adaptation approach from Daumé III (2007) to augment the feature space for FrameNet SRL with FrameNet example sentences; FitzGerald et al. (2015) and Hermann et al. (2014) adopt deep learning methods, including learning representations that may generalize better to unseen data, to present state-of-the-art results for FrameNet SRL. All of the former only use the already introduced split of the FrameNet fulltext corpus for testing, as does the long-time state-of-the-art system `Semafor` (Das et al., 2014). Out-of-domain evaluation is lacking, as are datasets that enable this kind of evaluation.

Frame identification Current state of the art in frame identification is the approach by Hermann et al. (2014), further referred to as `Hermann-14`, followed by the previous state-of-the-art model `Semafor` (Das et al., 2014).

The frame identification system of `Semafor` relies on an elaborate feature set based on syntactic and lexical features, using the WordNet hierarchy as a source of lexical information, and a label propagation-based approach to take unknown predicates into account. `Semafor` is not specifically designed for out-of-domain use: the WordNet coverage is limited, and the quality of syntactic parsing might drop when the system is applied to out-of-domain data, especially in case of non-standard user-generated texts.

`Hermann-14` uses distributed word representations augmented by syntactic information. General-purpose distributed word representations (such as `word2vec` (Mikolov et al., 2013) and `GloVe` (Pennington et al., 2014)) are beneficial for many NLP tasks: word representations are calculated on a large unlabeled corpus, and then used as input for high-level tasks for which training data is scarce, such as syntactic parsing, word sense disambiguation, and SRL. In the syntax-augmented representations of `Hermann-14`, a region of the input vector, a *container*, is reserved for each syntactic path that can connect predicates to their arguments. This container is populated with a corresponding argument word representation, if the argument on this path is found in the training data. `Hermann-14` uses the WSABIE algorithm (Weston et al., 2011) to map input and frame representations to a common latent space. WSABIE uses WARP loss and gradient-based updates to minimize the distance between the latent representations of the predicate target and the correct frame, while maximizing the distance to all the other irrelevant frames. During testing, cosine similarity is used to find the closest frame given the input. One advantage of this approach is that similar frames are positioned close to each other in the latent space which allows information to be shared between similar predicates and similar frames. This system is the current state-of-the-art for in-domain frame identification, but has not been applied in an out-of-domain setting.

3 Out-of-domain FrameNet test data

This section describes available in-domain and out-of-domain FrameNet test sets and the creation of YAGS, a new out-of-domain FrameNet test set.

FrameNet test sets FrameNet SRL is typically evaluated on **das-test**, the test set first introduced by Das and Smith (2011). It is a held-out set randomly sampled from the FrameNet 1.5 fulltext cor-

If [you]_{Grinder} have a [mortal and pestle]_{Grinding_instrument}, **grind**_{Grinding.head} **up**_{Grinding.satellite}
 [all the ingredients]_{Undergoer} [in the order above] _{Manner} [with it]_{Instrument}.

Figure 1: Example sentence from YAGS with multiword predicate and typo (*mortal* vs. *mortar*).

pus. While the FrameNet fulltext corpus contains data from various sources, we consider *das-test* an *in-domain* test set: all data sources of the test set are also represented in the training set.

There are two additional datasets from other domains that we use in our study on domain generalization: The **MASC** word sense sentences corpus contains FrameNet annotations for a lexical sample of roughly 100 lemmas from ANC (Passonneau et al., 2012). The Twitter-based dataset from Søggaard et al. (2015), henceforth **TW**, has some very distinctive properties: it does not provide a gold standard, but annotations by three annotators. This leads to a high variance in role annotations: the annotator TW_3 annotated only 82% of the number of roles annotated by TW_1 , see Table 1. Like Søggaard et al. (2015), we report SRL results as averages over the three annotations (TW_{-av}).

Table 1 shows statistics on these datasets. For **TW**, it displays the statistics for each annotator. The **TW** datasets are fairly small, containing only around 1,000 frame labels. The **MASC** dataset is of substantial size, but it constitutes a lexical sample and therefore a slightly artificial evaluation setup. There is another Twitter-based test set (Johannsen et al., 2015), which we do not use in our experiments, because it was created semi-automatically and is therefore of lower quality. We conclude that existing out-of-domain test sets for FrameNet SRL are insufficient, in particular for increasingly important domains like user-generated text, because available datasets are either small or of low quality.

YAGS: a new FrameNet test set based on user-generated text To address the need for new out-of-domain test datasets, we created **YAGS**, a new FrameNet-annotated evaluation dataset based on question-answer data from Yahoo! Answers (YA), a community-driven question-and-answer forum. The corpus is based on a random sample of 55 questions and their answers from the test split of the YA Manner Questions dataset used by Surdeanu et al. (2011) and published as part of the Yahoo! Webscope program (<https://webscope.sandbox.yahoo.com/>).

YAGS contains 1,415 sentences, 3,091 frame annotations, and 6,081 role annotations. Figure 1 shows a sentence from **YAGS** that demonstrates some non-standard properties of the user-generated question-answer data, such as typos (*mortal* instead of *mortar*). We publish the annotations as stand-off annotations to the original dataset.

Annotation study Each document was annotated by a two linguistically trained annotators provided with detailed guidelines and then curated by an experienced expert, all using WebAnno 2.0.0 (Yimam et al., 2014). Up to five predicates per sentence were pre-selected automatically based on lemma and POS, preferring verbal predicates to other POS, which leads to a larger proportion of verbs in **YAGS**. The annotation task was to identify the correct frame label for each predicate, if any, and then to identify the role spans as arguments and adjuncts of the frame, and to label them with the appropriate role. For reference, annotators accessed the FrameNet 1.5 definitions and examples with the FrameNet Explorer tool (www.clres.com/FNExplorer.html).

Inter-rater agreement for frame labels is Krippendorff’s $\alpha=0.76$; agreement for role labels given matching spans is $\alpha=0.62$, and Krippendorff’s α unitizing agreement for role spans is 0.7 – a good result for such a difficult task on user-generated text. Average pairwise F_1 agreement for frame labels is high at 0.96, higher than the 0.84 reported by Søggaard et al. (2015) for the **TW** sets. Our high frame agreement is a result of annotator experience and our elaborate annotation setup.

YAGS statistics and properties Table 1 presents dataset statistics for **YAGS** and the other test sets. Due to the predicate selection, **YAGS** contains a larger proportion of verbal predicates than the other sets, and has three times more frames and roles than **TW**, approximating the size of *das-test*. The proportion of core roles, roles that are obligatory for a frame and thus typically more frequent in datasets than non-core roles, in the out-of-domain test sets (**TW**, **YAGS**, **MASC**) is slightly smaller

data	s	f	a	n	v	r	cr
das-test	2,420	4,458	12	42	33	7,172	83
YAGS	1,415	3,091	5	18	75	6,081	74
MASC	8,444	7,226	25	42	33	11,214	78
TW ₁	236	1,085	10	47	40	1,704	77
TW ₂	236	1,027	11	46	39	1,614	79
TW ₃	236	1,038	11	47	39	1,399	89

Table 1: Text dataset statistics: sentences **s**; frames **f**; % of adjectives **a**, nouns **n** and verbs **v**; roles **r**, % of core roles **cr**. Subscripts for TW indicate the respective annotator.

compared to das-test. This goes along with a larger variance of roles in YAGS.

The user-generated aspect of YAGS manifests in spelling errors, and in the lack of punctuation and structure of the texts. The language is informal, but there are only few emoticons or other special words such as the hashtags typically found in tweets.

In the next section, we use the test sets from Table 1 to analyze the domain generalization capabilities of an open-source FrameNet SRL system.

4 Domain generalization capabilities of open-source FrameNet SRL

To analyze the domain generalization capabilities of contemporary open-source SRL, we ran the frame identification from *Semafor* (Das et al., 2014) with the enhanced role labeler from Kshirsagar et al. (2015), both trained on the in-domain das-train set, on the four test sets das-test, YAGS, TW, and MASC. The systems receive text annotated with predicate spans as input, which has become the standard in recent evaluations.

Evaluation script The *Semafor* evaluation script (Das et al., 2014) provides precision P, recall R, and F_1 scores for full SRL (SRL), and accuracy A for frame identification (frameId). Full SRL evaluation can be performed with and without using gold frames instead of predicted (auto) frames.

The script does not provide results on the role labeling (argument identification and labeling, roleId) alone: the scoring mechanism for *SRL/gold* also considers the by default correct gold frames. This is useful when comparing different SRL systems on the same test set, but not sufficient when 1) comparing role labeling performance on different test sets with a different ratio of frame labels to role labels (resulting from different annotation strategies), and 2) analyzing the contribution of frameId and roleId to full SRL performance across test sets.

data	frameId	roleId		SRL	
	auto	auto	gold	auto	gold
das-test	82.09	30.08	55.20	55.40	73.16
YAGS	59.62	18.60	56.99	37.22	72.58
MASC	39.52	19.46	51.74	29.05	71.08
TW-av	62.17	15.91	61.45	38.44	76.74

Table 2: *Semafor* performance on test sets in %: exact frameId A; then F_1 for roleId and SRL with system frames (auto) and gold frames (gold).

We therefore evaluate the output of the script to retain the original counts for role labels and compute scores on the role labeling proper (roleId). Moreover, there are two evaluation settings for frameId: exact frame match and partial frame match. We use the exact match setting that does not credit related frames and roles.

Results Table 2 presents scores for exact match frameId and for SRL and roleId with automatic frames (auto) and with gold frames (gold). For TW, the results are averaged over the number of annotators. According to column *SRL/auto*, we observe best *Semafor* performance for full SRL on das-test, results for the other test sets are at least 16 percentage points F_1 lower. This is mostly due to the worse frameId performance of *Semafor* on the new test sets, as shown in column *frameId*: frameId performance is at least 19 percentage points lower. This negatively affects roleId for the out-of-domain test sets (see column *roleId/auto*). *RoleId/auto* scores are also low on das-test, but higher than for the other sets.

When using gold frame labels, roleId and SRL performance improve for all test sets. As shown in columns *roleId/gold* and *SRL/gold*, the difference between in-domain and out-of-domain evaluation vanishes. Only MASC scores are still two points lower for full SRL than those for das-test. TW-av scores even surpass the in-domain scores.²

This shows how much FrameNet role labels are dependent on correct frame labels. Thus, it is crucial to improve the out-of-domain performance of frameId systems.

Domain dependence appears to be less of a problem for the role labeling step. The MASC dataset is the most difficult for both frameId and roleId. This is mostly a consequence of the lower training data coverage of MASC, as discussed below.

²Our TW-av results are not comparable to those from Sogaard et al. (2015) because their test setup includes predicate target identification and uses different evaluation metrics.

dataset	lemmas \notin		senses \notin	monosemous
	lexicon	das-train	das-train	\in das-train
das-test	2.59	9.99	14.03	53.99
YAGS	2.79	17.33	30.36	27.07
MASC	7.45	21.72	51.25	23.51
TW ₁	1.01	17.51	36.06	26.73
TW ₂	1.27	17.91	51.25	27.07
TW ₃	1.25	17.24	35.65	27.17

Table 3: Training data coverage of test sets in %. Sense is a combination of predicate lemma, POS and frame; lexicon refers to the `SemaFor` lexicon.

Analysis In our study, it became clear that domain dependence is crucial to the frame identification step in SRL. The lower scores for the out-of-domain test sets can be a result of different domain-specific predicate-frame distributions, or a lack of coverage of the domain in the training data.

To get a better understanding of these phenomena, we compared detailed statistics of the different test sets, cf. Table 3. Das-test has the largest predicate coverage and contains a lot of monosemous predicates, which boosts the overall performance. The occurrence of fewer monosemous predicates is expected for the lexical sample dataset MASC, but might indicate a domain preference for polysemous predicates in the YAGS and TW datasets.

The percentage of unseen predicates (lemmas \notin das-train) is slightly higher for the user-generated test sets than for das-test, and much higher for MASC. This is mirrored in the lower frameId performance for MASC compared to the other test sets, and the slightly higher performance of TW-av and YAGS. Not all errors can be explained by insufficient training data coverage, which indicates that domain effects occur for the out-of-domain sets.

To support this assumption, we performed a detailed error analysis on the misclassified instances for all test sets. We compute the proportion of wrongly classified instances with unseen predicates, predicates that do not occur in the training set. For MASC, the majority of the errors, 68%, are based on unseen predicates, while the number ranges between 37% and 43% for the other test sets, i.e. 37% for TW, 39% for das-test and 43% for YAGS. This shows that training data coverage is a bigger issue for MASC than for the other test sets. The proportions of in-train errors for YAGS and TW-av are similar to das-test. Together with the fact that overall proportion of errors is still much higher for the user-generated test sets YAGS and TW-av, this further supports our hypothesis of domain effects

for YAGS and TW-av. Manual analysis furthermore shows that there are differences in frequently confused frames between the in-domain das-test and out-of-domain YAGS and TW-av.

In the next section, we study new methods to improve out-of-domain frame identification.

5 Frame identification with distributed word representations

Given a predicate and a set of frames associated with this predicate, a frame identification system has to choose the correct frame based on the context. In this section we introduce our frame identification method and compare it to the state of the art in both in-domain and out-of-domain settings.

Our system SimpleFrameId We developed a straightforward approach to frame identification based on distributed word representations, and were surprised to find that this simple model achieves results comparable to the state-of-the-art system, `Hermann-14`. Our initial attempts to replicate `Hermann-14`, which is not publicly available, revealed that the container-based input feature space is very sparse: there exist many syntactic paths that can connect a predicate to its arguments, but a predicate instance rarely has more than five arguments in the sentence. So by design the input representation bears no information in most of its path containers. Moreover, `Hermann-14` makes heavy use of automatically created dependency parses, which might decline in quality when applied to a new domain. We demonstrate that our simple system achieves competitive in-domain and out-of-domain performance.

Our system, called `SimpleFrameId`, is specified as follows: given the lexicon L , the vector space vsm and the training data, our goal is to predict the frame f given the sentence S and the predicate p . From the machine learning perspective, the lexicon and the vector space are external resources. The lexicon contains associations between predicates and frames, and we further denote the set of frames available for a predicate as $L(p)$. The vector space provides a pre-defined dense vector representation $vsm(w)$ for each word w . In our case vsm is a simple word lookup function, since we do not modify our word representations during training.

From the sentence we extract the context representation, $x_c = \frac{\sum_{w \in C} vsm(w)}{|C|}$. We experiment with two kinds of contexts: `SentBOW` includes all

the words in the sentence, i.e. $C = S$, DepBOW considers the dependency parse of the sentence and only includes direct dependents of the predicate, $C = \text{dep}(p, S)$. As for the predicate, the plain embedding from the source vector space model is used, $x_p = \text{vsm}(p)$. A simple concatenation of x_c and x_p serves as input to the disambiguation classifier D , which outputs weights $D(x_c, x_p, f)$ for each frame known to the system $f \in L$. Note that the classifier itself is agnostic to the predicate’s part of speech and exact lemma and only relies on the word representations from the *vsm*. We experiment with two different classification methods: one is a two-layer neural network D_{NN} , the other one is D_{WSB} , which follows the line of Hermann-14 and learns representations for frames and predicates in the same latent space using the WSABIE algorithm.³ Hyperparameters are tuned on the development sets *das-dev* and *YAGS-dev* (sampled from *YAGS*); we test on the remaining 2,093 instances in *YAGS-test*.

Lexicon-based filtering In the testing stage, the classifier outputs weights for all the frames available in the lexicon, and the best-scoring frame is selected, $f \leftarrow \text{argmax}_{f \in L} D(x_c, x_p, f)$. Since the lexicon specifies available frames for each lexical unit (i.e. lemma and POS), additional filtering can be performed, which limits the search only to the available frames, $f \leftarrow \text{argmax}_{f \in L(p)} D(x_c, x_p, f)$. If the predicate is *unknown* to the lexicon, $p \notin L$, the overall best-scoring frame is chosen. If the target has only one entry in the lexicon, it’s declared unambiguous and the frame is assigned directly.

Despite being common, this setup has several flaws that can obscure the differences between systems in the testing stage. As we showed in Section 4, the FrameNet lexicon has coverage issues when applied to new domains. Neither the predicate list nor the frame associations are guaranteed to be complete, and hence the total results are highly determined by the lexicon coverage.⁴ To take this into account, we also perform evaluation in the *no-lexicon* setting, where frames are assigned directly by the classifier and no lexicon-based fil-

³In our implementation, we use the LightFM package (Kula, 2015) with the WARP option for hybrid matrix factorization.

⁴A justification for this can also be found in Hermann et al. (2014): the difference in Hermann-14 accuracy when switching from the *Semafor* lexicon to the full lexicon is comparable to the difference between *Semafor* and Hermann-14 when evaluated on the same lexicon.

system	total	ambig	no-lex
DataBaseline	79.09	70.68	2.21
LexiconBaseline	79.05	56.62	2.21
Semafor*	83.60	69.19	-
Hermann-14* (best)	88.41	73.10	-
WSB+SentBOW	84.46	67.56	72.05
WSB+DepBOW	85.69	69.93	71.21
NN+SentBOW	87.63	73.80	77.49
NN+DepBOW	87.53	73.58	76.51

Table 4: In-domain system comparison on *das-test*, * denotes results from Hermann et al. (2014); **ambig**: evaluation on ambiguous predicates; **no-lex**: system without lexicon filter.

tering is performed. We find that our frame identification system performs surprisingly well in this setting, and we encourage the *no-lexicon* performance to be additionally reported in the future, since it better reflects the frame identification quality and smoothens the effect of lexicon coverage.

Baselines We employ two majority baseline models for comparison. The *DataBaseline* assigns frames based on how often a frame is evoked by the given predicate. This corresponds to the most frequent sense baseline in word sense disambiguation (WSD). The frames available for predicates are obtained by scanning the training data. The *LexiconBaseline* calculates overall frame counts first (i.e. how often a frame appears in the training data in general), and, given the predicate, selects the overall most frequent frame among the ones available for this predicate. We expect this baseline to better handle the cases when limited data is available for a given predicate sense.

Experiments In our experiments, we generate the lexicon L in the same way as in Hermann-14, by scanning the “frames” folder of the FrameNet 1.5 distribution. For the external vector space model *vsm* we use dependency-based word embeddings from Levy and Goldberg (2014).

In-domain performance We report the performance of our system in the in-domain setting to compare to the state-of-the-art results from Hermann-14.⁵ We train our system on *das-train* and test it on *das-test* using the full FrameNet lexicon. When available, we report the *no-lexicon* scores as well. As Table 4 shows, our system out-

⁵Based on the errata version of Hermann et al. (2014) in <http://www.aclweb.org/anthology/P/P14/P14-1136v2.pdf>

system	das-test	YAGS	MASC	TW-av
DataBaseline	79.09	52.27	43.85	47.68
LexiconBaseline	79.05	50.02	36.86	55.40
Semafor	82.09	60.01	39.52	62.17
WSB+SentBOW	84.46	59.68	54.90	66.84
WSB+DepBOW	85.69	61.50	54.56	67.14
NN+SentBOW	87.63	62.03	53.73	68.67
NN+DepBOW	87.53	62.51	55.09	67.76

Table 5: Out-of-domain frameId, total accuracy. Semafor scores calculated during our own experiments; YAGS results on YAGS-test.

performs Semafor and performs on par with the results reported for Hermann-14. One interesting observation is that our systems perform almost as well in the no-lexicon setting as the DataBaseline, which has access to the lexicon, in the total setting. To our surprise, the WSABIE-based frame identification did not yield a consistent improvement in-domain, compared to the simple NN-based approach. We also observe that in many cases the SentBOW representation performs on par with the DepBOW, while requiring significantly less data preprocessing: SentBOW only uses tokenization, whereas DepBOW relies on lemmatization, POS-tagging, and dependency parsing. We attribute this effect to the fact that SentBOW provides more context information than the sparse, dependency-filtered DepBOW.

Out-of-domain performance We also investigate how well the systems perform in the out-of-domain setting. Table 5 summarizes the results. Each of the systems was trained on *das-train* and tested on a variety of test sets. As we can see, our systems outperform Semafor for all datasets. The YAGS dataset is the only dataset on which we do not strongly outperform Semafor. We attribute this to the complexity of the YAGS dataset that contains a high proportion of verbs.

Overall out-of-domain performance stays behind the F_1 -agreement observed for the human annotators for TW and YAGS, which shows that there is a large margin for improvement. Corresponding scores for in-domain data are not available.

Error analysis To further investigate the performance of our system in the out-of-domain setup we analyse statistics on the errors made by the system variant NN+SentBOW.

The system’s wrong predictions are affected by the lexicon in two ways. First, if the predicate is

not listed in the lexicon (unknown), the system has to choose among all frames. As we have shown before, the quality of predictions for unknown predicates is generally lower. The second case is when the predicate *is* listed in lexicon (so it is not unknown), but the correct frame is not associated with this predicate. We further refer to this class of errors as *unlinked*. For unlinked predicates, the system is restricted to the set of frames provided by the lexicon, and by design has no means to select the right frame for a given predicate occurrence.

The unlinked-predicate issue points to a major design flaw in the standard frameId architecture. Although choosing among frames defined in the lexicon provides a quality boost, it also renders many instances intractable for the system, if the lexicon coverage is incomplete. As Table 6 shows, unknown and unlinked predicates are almost non-present in the in-domain case, but are a major source of errors in the out-of-domain case and even might be responsible for the majority of errors occurring due to domain shift (see MASC). It is important to point out that there is still no guarantee that these would be classified correctly once the missing linking information is available in the lexicon. However, if the correct frame is not listed among the frames available for the predicate, the misclassification is inevitable.

A more detailed analysis of the errors made by the system shows that the majority of false predictions for known and linked predicates are due to the domain differences in word usage. For example, the predicate **window** was assigned the frame *Connecting_architecture* instead of the correct frame *Time_period_of_action* in the following sentence:

“No effect of anesthetic protocol on IOP during a 12 minute measurement [**window**].”

This problem is also relevant in generic WSD (Agirre et al., 2010) and benefits from the same solutions, for instance adapting embeddings to a particular domain (Taghipour and Ng, 2015) and efficient use of embeddings (Iacobacci et al., 2016).

Another major source of errors are subtle syntactic and semantic differences between frames which are hard to resolve on the sentence level (e.g. distinguishing between *Similarity* and *Identity* for the predicate **different**). This could be addressed by incorporating subcategorization information and document context into the disam-

dataset	% errors			accuracy loss	
	unk	unl	Σ	unk∪unl	total
test-das	0.83	0.66	1.49	0.18	-
YAGS-test	3.76	13.05	16.81	6.40	25.60
MASC	12.15	33.70	45.85	24.03	33.90
TW-avg	10.40	9.68	20.08	6.31	18.96

Table 6: Error sources for NN+Dep; **unk** is the percentage of unknown and **unl** is the percentage of unlinked predicates among misclassified instances.

biguation model, which has been proposed in recent work in FrameNet SRL, see e.g. Hermann et al. (2014) and Roth and Lapata (2015).

To further explore the impact of user-generated text, we applied word-processor spelling correction to YAGS and tested our systems on the corrected set. The results do not change significantly, which indicates that a) our distributed representations provide enough information to classify also noisy user-generated text, and b) frameId errors cannot be attributed to preprocessing problems at large scale.

6 Discussion and outlook

Our analysis in Section 4 shows that domain adaptation is mainly required for the frameId step of FrameNet SRL. Unlike in PropBank SRL, in FrameNet SRL there is no significant performance drop for roleId once correct frames are available. The number of available roles given the correct frame is lower, on average 10, which reduces the complexity of the roleId task.

In Section 5 we introduced a simple, yet efficient frame identification method and evaluated it on in-domain and out-of-domain data. The method achieves competitive in-domain results, and outperforms the best available open-source system in out-of-domain accuracy. We also observe that our system performs well in the newly introduced `no-lexicon` evaluation setting, where no lexicon-based filtering is applied.

We identified a major issue in the standard frameId architecture: shifting to a new domain might render the predicate-frame associations in the FrameNet lexicon incomplete, which leads to errors for a standard classifier trained on in-domain data. One could optimize a frameId system to work in the `no-lexicon` setting which does not rely on the lexicon knowledge at all. However, in this setting the classification results are currently lower. Manually or automatically increasing both predicate and predicate-frame association coverage of

the FrameNet lexicon could help, and we suggest investigating this line of research in future work.

While our method achieves state-of-the-art results on out-of-domain data, overall results are still significantly lower than the human performance observed for YAGS and TW, which shows that there is large room for improvement. Some further benefits could be gained from combining the WSABIE and NN-based classification, using advanced context representations, e.g. *context2vec* (Melamud et al., 2016) and incorporating syntactic information into the model. The out-of-domain performance could be further improved by adapting word representations to a new domain.

A direct comparison to the `Hermann-14` system in the out-of-domain setup would shed some more light on the properties of the task affecting the out-of-domain performance. On the one hand, we expect `Hermann-14` to perform worse due to its heavy reliance on syntactic information, which might decline in quality when moved to a new domain; on the other hand, the WSABIE-based classification might smoothen this effect. We make our dataset publicly available to enable comparison to related work.⁶

7 Conclusion

Domain dependence is a well-known issue for supervised NLP tasks such as FrameNet SRL. To the best of our knowledge, there is no recent study of the domain dependence of FrameNet SRL, also prohibited by a lack of appropriate datasets.

To address this problem, we 1) present the first comprehensive study of the domain generalization performance of the open-source `SemaFor` system on several diverse benchmark sets. As a prerequisite, we introduce YAGS, a new, substantially sized test set in the domain of user-generated question-and-answer text. We find that the major bottleneck for out-of-domain FrameNet SRL is the frame identification step; we 2) explore a promising way to improve out-of-domain frame identification, i.e. using distributed word representations. Our simple frame identification system based on distributed word representations achieves higher scores for out-of-domain frame identification than previous systems and approaches state-of-the-art results in-domain. To support reproducibility of our results, we publish the YAGS test set annotations and our frame identification system for research purposes.

⁶www.ukp.tu-darmstadt.de/ood-fn-srl

Acknowledgements

This work was supported by FAZIT-Stiftung and by the German Research Foundation (DFG) through grant GU 798/18-1 (QAEduInf) and the research training group “Adaptive Preparation of Information from Heterogeneous Sources” (AIPHES, GRK 1994/1). We thank Orin Hargraves and our annotators for their excellent work on the annotation study, Dr. Richard Eckart de Castilho for support regarding WebAnno, as well as Dr. Judith Eckle-Kohler and the anonymous reviewers for their comments on earlier versions of this paper.

References

- Eneko Agirre, Oier López de Lacalle, Christiane Fellbaum, Shu-Kai Hsieh, Maurizio Tesconi, Monica Monachini, Piek Vossen, and Roxanne Segers. 2010. SemEval-2010 Task 17: All-Words Word Sense Disambiguation on a Specific Domain. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 75–80. Association for Computational Linguistics.
- Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 Task 19: Frame Semantic Structure Extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, Czech Republic, June. Association for Computational Linguistics.
- Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling Biological Processes for Reading Comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar. Association for Computational Linguistics.
- John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.
- Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Danilo Croce, Cristina Giannone, Paolo Annesi, and Roberto Basili. 2010. Towards open-domain semantic role labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 237–246, Uppsala, Sweden, July. Association for Computational Linguistics.
- Dipanjan Das and Noah A. Smith. 2011. Semi-Supervised Frame-Semantic Parsing for Unknown Predicates. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1435–1444, Portland, Oregon, USA.
- Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic Frame-Semantic Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 948–956, Los Angeles, California. Association for Computational Linguistics.
- Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.
- Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.
- Katrin Erk and Sebastian Padó. 2006. SHALMANESER – A Toolchain For Shallow Semantic Parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, volume 6, pages 527–532, Genoa, Italy. ELRA.
- Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International journal of lexicography*, 16(3):235–250.
- Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal, September. Association for Computational Linguistics.
- Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.
- Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1458, Baltimore, Maryland, June. Association for Computational Linguistics.

- Fei Huang and Alexander Yates. 2010. Open-domain semantic role labeling by modeling word spans. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 968–978, Uppsala, Sweden, July. Association for Computational Linguistics.
- Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany, August. Association for Computational Linguistics.
- Anders Johannsen, Héctor Martínez Alonso, and Anders Søgaard. 2015. Any-language frame-semantic parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2062–2066, Lisbon, Portugal, September. Association for Computational Linguistics.
- Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 393–400, Manchester, UK, August. Coling 2008 Organizing Committee.
- Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 218–224, Beijing, China, July. Association for Computational Linguistics.
- Maciej Kula. 2015. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015)*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21, Vienna, Austria, September. CEUR-WS.org.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308. The Association for Computer Linguistics.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 51–61.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS '13)*, pages 3111–3119, Lake Tahoe, Nevada, USA.
- Alexis Palmer and Caroline Sporleder. 2010. Evaluating FrameNet-style semantic parsing: the role of coverage gaps in FrameNet. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 928–936, Beijing, China, August.
- Rebecca J. Passonneau, Collin F. Baker, Christiane Fellbaum, and Nancy Ide. 2012. The MASC Word Sense Corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3025–3030, Istanbul, Turkey.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.
- Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460.
- Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2010. FrameNet II: Extended Theory and Practice. Technical report, ICSI, University of California, Berkeley.
- Anders Søgaard, Barbara Plank, and Héctor Martínez Alonso. 2015. Using Frame Semantics for Knowledge Extraction from Twitter. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2447–2452, Austin, Texas, USA.
- Anders Søgaard. 2013. Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies*, 6(2):1–103.
- Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.
- Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383.

- Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, Denver, Colorado, May–June. Association for Computational Linguistics.
- Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: Scaling Up to Large Vocabulary Image Annotation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI'11*, pages 2764–2770, Barcelona, Catalonia, Spain. AAAI Press.
- Haitong Yang, Tao Zhuang, and Chengqing Zong. 2015. Domain adaptation for syntactic and semantic dependency parsing using deep belief networks. *Transactions of the Association for Computational Linguistics*, 3:271–282.
- Seid Muhie Yimam, Richard Eckart de Castilho, Iryna Gurevych, and Chris Biemann. 2014. Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno. In Kalina Bontcheva and Zhu Jingbo, editors, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 91–96, Stroudsburg, PA 18360, USA. Association for Computational Linguistics.

TDParse: Multi-target-specific sentiment recognition on Twitter

Bo Wang Maria Liakata Arkaitz Zubiaga Rob Procter

Department of Computer Science

University of Warwick

Coventry, UK

{bo.wang, m.liakata, a.zubiaga}@warwick.ac.uk

Abstract

Existing target-specific sentiment recognition methods consider only a single target per tweet, and have been shown to miss nearly half of the actual targets mentioned. We present a corpus of UK election tweets, with an average of 3.09 entities per tweet and more than one type of sentiment in half of the tweets. This requires a method for multi-target specific sentiment recognition, which we develop by using the context around a target as well as syntactic dependencies involving the target. We present results of our method on both a benchmark corpus of single targets and the multi-target election corpus, showing state-of-the-art performance in both corpora and outperforming previous approaches to multi-target sentiment task as well as deep learning models for single-target sentiment.

1 Introduction

Recent years have seen increasing interest in mining Twitter to assess public opinion on political affairs and controversial issues (Tumasjan et al., May 2010; Wang et al., 2012) as well as products and brands (Pak and Paroubek, 2010). Opinion mining from Twitter is usually achieved by determining the overall sentiment expressed in an entire tweet. However, inferring the sentiment towards specific targets (e.g. people or organisations) is severely limited by such an approach since a tweet may contain different types of sentiment expressed towards each of the targets mentioned. An early study by Jiang et al. (2011) showed that 40% of classification errors are caused by using tweet-level approaches that are independent of the target. Consider the tweet:

*“I will b voting 4 **Greens** ... 1st reason:
2 remove 2 party alt. of **labour** or **conservative** every 5 years. 2nd: **fracking**”*

The overall sentiment is positive but there is a negative sentiment towards “labour”, “conservative” and “fracking” and a positive sentiment towards “Greens”. Examples like this are common in tweets discussing topics like politics. As has been demonstrated by the failure of election polls in both referenda and general elections (Burnap et al., 2016), it is important to understand not only the overall mood of the electorate, but also to distinguish and identify sentiment towards different key issues and entities, many of which are discussed on social media on the run up to elections.

Recent developments on target-specific Twitter sentiment classification have explored different ways of modelling the association between target entities and their contexts. Jiang et al. (2011) propose a rule-based approach that utilises dependency parsing and contextual tweets. Dong et al. (2014), Tang et al. (2016a) and Zhang et al. (2016) have studied the use of different recurrent neural network models for such a task but the gain in performance from the complex neural architectures is rather unclear¹

In this work we introduce the multi-target-specific sentiment recognition task, building a corpus of tweets from the 2015 UK general election campaign suited to the task. In this dataset, target entities have been semi-automatically selected, and sentiment expressed towards multiple target entities as well as high-level topics in a tweet have been manually annotated. Unlike all existing studies on target-specific Twitter sentiment analysis, we move away from the assumption that

¹They have yet to show a clear out-performance on a benchmarking dataset and our multi-target corpus, possibly because they usually require large amount of training data.

each tweet mentions a single target; we introduce a more realistic and challenging task of identifying sentiment towards multiple targets within a tweet. To tackle this task, we propose TDParse, a method that divides a tweet into different segments building on the approach introduced by Vo and Zhang (2015). TDParse exploits a syntactic dependency parser designed explicitly for tweets (Kong et al., 2014), and combines syntactic information for each target with its left-right context.

We evaluate and compare our proposed system both on our new multi-target UK election dataset, as well as on the benchmarking dataset for single-target dependent sentiment (Dong et al., 2014). We show a clear state-of-the-art performance of TDParse over existing approaches for tweets with multiple targets, which encourages further research on the multi-target-specific sentiment recognition task.²

2 Related Work: Target-dependent Sentiment Classification on Twitter

The 2015 Semeval challenge introduced a task on target-specific Twitter sentiment (Rosenthal et al., 2015) which most systems (Boag et al., 2015; Plotnikova et al., 2015) treated in the same way as tweet level sentiment. The best performing system in the 2016 Semeval Twitter challenge subtask B (Nakov et al., 2016), named Tweester, also performs on tweet level sentiment classification. This is unsurprising since tweets in both tasks only contain a single predefined target entity and as a result often a tweet-level approach is sufficient. An exception to tweet level approaches for this task, showing promise, is Townsend et al. (2015), who trained a SVM classifier for tweet segmentation, then used a phrase-based sentiment classifier for assigning sentiment around the target. The Semeval aspect-based sentiment analysis task (Pontiki et al., 2015; Pateria and Choubey, 2016) aims to identify sentiment towards entity-attribute pairs in customer reviews. This differs from our goal in the following way: both the entities and attributes are limited to a predefined inventory of limited size; they are aspect categories reflected in the reviews rather than specific targets, while each review only has one target entity, e.g. a laptop or a restaurant. Also sentiment classification in formal text such as product reviews

is very different from that in tweets. Recently Vargas et al. (2016) analysed the differences between the overall and target-dependent sentiment of tweets for three events containing 30 targets, showing many significant differences between the corresponding overall and target-dependent sentiment labels, thus confirming that these are distinct tasks.

Early work tackling target-dependent sentiment in tweets (Jiang et al., 2011) designed target-dependent features manually, relying on the syntactic parse tree and a set of grammar-based rules, and incorporating the sentiment labels of related tweets to improve the classification performance. Recent work (Dong et al., 2014) used recursive neural networks and adaptively chose composition functions to combine child feature vectors according to their dependency type, to reflect sentiment signal propagation to the target. Their data-driven composition selection approach relies on the dependency types as features and a small set of rules for constructing target-dependent trees. Their manually annotated dataset contains only one target per tweet and has since been used for benchmarking by several subsequent studies (Vo and Zhang, 2015; Tang et al., 2016a; Zhang et al., 2016). Vo and Zhang (2015) exploit the left and right context around a target in a tweet and combine low-dimensional embedding features from both contexts and the full tweet using a number of different pooling functions. Despite not fully capturing semantic and syntactic information given the target entity, they show a much better performance than Dong et al. (2014), indicating useful signals in relation to the target can be drawn from such context representation. Both Tang et al. (2016a) and Zhang et al. (2016) adopt and integrate left-right target-dependent context into their recurrent neural network (RNN) respectively. While Tang et al (2016a) propose two long short-term memory (LSTM) models showing competitive performance to Vo and Zhang (2015), Zhang et al (2016) design a gated neural network layer between the left and right context in a deep neural network structure but require a combination of three corpora for training and evaluation. Results show that conventional neural network models like LSTM are incapable of explicitly capturing important context information of a target (Tang et al., 2016b). Tang et al. (2016a) also experiment with adding attention layers for LSTM but

²The data and code can be found at <https://goo.gl/S2T1GO>

fail to achieve competitive results possibly due to the small training corpus.

Going beyond the existing work we study the more challenging task of classifying sentiment towards multiple target entities within a tweet. Using the syntactic information drawn from tweet-specific parsing, in conjunction with the left-right contexts, we show the state-of-the-art performance in both single and multi-target classification tasks. We also show that the tweet level approach that many sentiment systems adopted in both Semeval challenges, fail to capture all target-sentiments in a multi-target scenario (Section 5.1).

3 Creating a Corpus for Target Specific Sentiment in Twitter

We describe the design, collection and annotation of a corpus of tweets about the 2015 UK election.

3.1 Data Harvesting and Entity Recognition

We collected a corpus of tweets about the UK elections, as we wanted to select a political event that would trigger discussions on multiple entities and topics. Collection was performed through Twitter’s streaming API and tracking 14 hashtags³. Data harvesting was performed between 7th February and 30th March 2015. This led to the collection of 712k tweets, from which a subset was sampled for manual annotation of target-specific sentiment. We also created a list of 438 topic keywords relevant to 9 popular election issues⁴ for data sampling. The initial list of 438 seed words provided by a team of journalists was augmented by searching for similar words within a vector space on the basis of cosine similarity. Keywords are used both in order to identify thematically relevant tweets and also targets. We also consider named entities as targets.

Sampling of tweets was performed by removing retweets and making sure each tweet contained at least one topic keyword from one of the 9 election issues, leading to 52,190 highly relevant tweets. For the latter we ranked tweets based on a “similarity” relation, where “similarity” is measured as a function of content overlap (Mihalcea, 2004). Formally, given a tweet S_i being represented by

³#ukelection2015, #ge2015, #ukge2015, #ukgeneralelection2015, #bbcqt, #bbcsp, #bbcdp, #marrshow, #generalelection2015, #ge15, #generalelection, #electionuk, #ukelection and #electionuk2015

⁴EU and immigration, economy, NHS, education, crime, housing, defense, public spending, environment and energy

the set of N words that appear in the tweet: $S_i = W_i^1, W_i^2, \dots, W_i^N$ and our list of curated topic keywords T , the ranking function is defined as:

$$\log(|S_i|) * |W_i \in S_i \cap W_i \in T| \quad (1)$$

where $|S_i|$ is the total number of words in the tweet; unlike Mihalcea (2004) we prefer longer tweets. We used exact matching with flexibility on the special characters at either end. TF-IDF normalisation and cosine similarity were then applied to the dataset to remove very similar tweets (empirically we set the cosine similarity threshold to 0.6). We also collected all external URLs mentioned in our dataset and their web content throughout the data harvesting period, filtering out tweets that only contain an external link or snippets of a web page. Finally we sampled 4,500 top-ranked tweets keeping the representation of tweets mentioning each election issue proportionate to the original dataset.

For annotation we considered sentiment towards two types of targets: entities and topic keywords. Entities were processed in two ways: firstly, named entities (people, locations, and organisations) were automatically annotated by combining the output of Stanford Named Entity Recognition (NER) (Finkel et al., 2005), NLTK NER (Bird, 2006) and a Twitter-specific NER (Ritter et al., 2011). All three were combined for a more complete coverage of entities mentioned in tweets and subsequently corrected by removing wrongly marked entities through manual annotation. Secondly, to make sure we covered all key entities in the tweets, we also matched tweets against a manually curated list of 7 political-party names and added users mentioned therein as entities. The second type of targets matched the topic keywords from our curated list.

3.2 Manual Annotation of Target Specific Sentiment

We developed a tool for manual annotation of sentiment towards the targets (i.e. entities and topic keywords) mentioned in each tweet. The annotation was performed by nine PhD-level journalism students, each of them annotating approximately a ninth of the dataset, i.e. 500 tweets. Additionally, they annotated a common subset of 500 tweets consist of 2,197 target entities, which was used to measure inter-annotator agreement (IAA). An-

Annotation of Target-Specific Tweet Sentiment

The screenshot shows a web-based annotation tool. At the top, there is a tab labeled 'Entities'. Below it, the instruction reads: 'Sentiment of the tweet towards the highlighted keyword(s):'. The main text of the tweet is: 'Ah so I compiled an analysis article on the lack of **defence** in #GE2015 and then **Ed Balls** drops this on me today. Cheers **Ed**'. The words 'defence', 'Ed Balls', and 'Ed' are highlighted in bold. Below each highlighted word, there are three sentiment icons: a smiley face (positive), a neutral face (neutral), and a sad face (negative), followed by an 'X' icon for 'does not apply'. Below the tweet text, there are three input fields for 'Additional entity #1:', 'Additional entity #2:', and 'Additional entity #3:'. Each field has a text input box and the same three sentiment icons and an 'X' icon to its right.

Figure 1: Annotation tool for human annotation of target specific sentiment analysis

notators were shown detailed guidelines⁵ before taking up the task, after which they were redirected to the annotation tool itself (see Figure 1).

Tweets were shown to annotators one by one, and they had to complete the annotation of all targets in a tweet to proceed. The tool shows a tweet with the targets highlighted in bold. Possible annotation actions consisted in: (1) marking the sentiment for a target as being positive, negative, or neutral, (2) marking a target as being mistakenly highlighted (i.e. ‘doesnotapply’) and hence removing it, and (3) highlighting new targets that our preprocessing step had missed, and associating a sentiment value with them. In this way we obtained a corrected list of targets for each tweet, each with an associated sentiment value.

We measure inter-annotator agreement in two different ways. On the one hand, annotators achieved $\kappa = 0.345$ ($z = 92.2, p < 0.0001$) (fair agreement)⁶ when choosing targets to be added or removed. On the other hand, they achieved a similar score of $\kappa = 0.341$ ($z = 77.7, p < 0.0001$) (fair agreement) when annotating the sentiment of the resulting targets. It is worth noting that the sentiment annotation for each target also involves choosing among not only positive/negative/neutral but also a fourth category ‘doesnotapply’. The resulting dataset contains 4,077 tweets, with an average of 3.09 entity mentions (targets) per tweet. As many as 3,713 tweets have more than a single entity mention (target) per tweet, which makes the task different from 2015 Semeval 10 subtask C (Rosenthal et al., 2015) and a target-dependent benchmarking dataset of Dong et al. (2014) where each tweet has only one target annotated and thus

one sentiment label assigned. The number of targets in the 4,077 tweets to be annotated originally amounted to 12,874. However, the annotators un-highlighted 975 of them, and added 688 new ones, so that the final number of targets in the dataset is 12,587. These are distributed as follows: 1,865 are positive, 4,707 are neutral, and 6,015 are negative. This distribution shows the tendency of a theme like politics, where users tend to have more negative opinions. This is different from the Semeval dataset, which has a majority of neutral sentiment. Looking at the annotations provided for different targets within each tweet, we observe that 2,051 tweets (50.3%) have all their targets consistently annotated with a single sentiment value, 1,753 tweets (43.0%) have two different sentiments, and 273 tweets (6.7%) have three different sentiment values. These statistics suggest that providing a single sentiment for the entire tweet would not be appropriate in nearly half of the cases confirming earlier observations (Jiang et al., 2011).

We also labelled each tweet containing one or more topics from the 9 election issues, and asked the annotators to mark the author’s sentiment towards the topic. Unlike entities, topics may not be directly present in tweets. We compare topic sentiment with target/entity sentiment for 3963 tweets from our dataset adopting the approach by Vargas et al. (2016). Table 1 reports the individual $c(s_{target})$, $c(s_{topic})$ and joint $c(s_{target}, s_{topic})$ distributions of the target/entity s_{target} and topic s_{topic} sentiment. While s_{target} and s_{topic} report how often each sentiment category occurs in the dataset, the joint distribution $c(s_{target}, s_{topic})$ (the inner portions of the table) shows the discrepancies between target and topic sentiments. We observe marked differences between the two sentiment labels. For example it shows the topic sentiment is more neutral (1438.7 vs. 1104.1) and less negative (1930.7 vs. 2285.5) than the target sen-

⁵This guidelines can be found along with our released corpus: <https://goo.gl/CjuHzd>

⁶We report the strength of agreement using the benchmarks by Landis and Koch (1977) for interpreting Fleiss’ kappa.

timent. There is also a number of tweets expressing neutrality towards the topics mentioned but polarised sentiment towards targets (i.e. we observe $c(s_{topic} = neu \cap s_{targets} = neg) = 258.6$ also $c(s_{topic} = neu \cap s_{targets} = pos) = 101.4$), and vice versa. This emphasises the importance of distinguishing target entity sentiment not only on the basis of overall tweet sentiment but also in terms of sentiment towards a topic.

$c(s_{target}, s_{topic})$		s_{topic}			$c(s_{topic})$
		negative	neutral	positive	
s_{target}	negative	1553.9	258.6	118.3	1930.9
	neutral	557.6	744.1	137.0	1438.7
	positive	174.0	101.4	318.1	593.5
$c(s_{target})$		2285.5	1104.1	573.4	3963.0

Table 1: Individual $c(s_{target})$, $c(s_{topic})$ and joint $c(s_{target}, s_{topic})$ distributions of sentiments

4 Developing a state-of-the-art approach for target-specific sentiment

4.1 Model development for single-target benchmarking data

Firstly we adopt the context-based approach by Vo and Zhang (2015), which divides each tweet into three parts (left context, target and right context), and where the sentiment towards a target entity results from the interaction between its left and right contexts. Such sentiment signal is drawn by mapping all the words in each context into low-dimensional vectors (i.e. word embeddings), using pre-trained embedding resources, and applying neural pooling functions to extract useful features. Such context set-up does not fully capture the syntactic information of the tweet and the given target entity, and by adding features from the full tweet (as done by Vo and Zhang (2015)) interactions between the left and right context are only implicitly modeled. Here we use a syntactic dependency parser designed explicitly for tweets (Kong et al., 2014) to find the syntactically connected parts of the tweet to each target. We then extract word embedding features from these syntactically dependent tokens $[D_1, \dots, D_n]$ along its dependency path in the parsing tree to the target⁷, as well as from the left-target-right contexts (i.e. $L - T - R$). Feature vectors generated from different contexts are concatenated into a final feature

⁷Empirically the proximity/location of such syntactic relations have not made much difference when used in feature weighting and is thus ignored.

vector as shown in (2), where $P(X)$ presents a list of k different pooling functions on an embedding matrix X . Not only does this proposed framework make the learning process efficient without labor intensive manual feature engineering and heavy architecture engineering for neural models, it has also shown that complex syntactic and semantic information can be effectively drawn by simply concatenating different types of context together without the use of deep learning (other than pre-trained word embeddings).

$$F = [P(D), P(L), P(T), P(R)]; \quad (2)$$

with $P(X) = [f_1(X), \dots, f_k(X)]$

Data set: We evaluate and compare our proposed system to the state-of-the-art baselines on a benchmarking corpus (Dong et al., 2014) that has been used by several previous studies (Vo and Zhang, 2015; Tang et al., 2016a; Zhang et al., 2016). This corpus contains 6248 training tweets and 692 testing tweets with a sentiment class balance of 25% negative, 50% neutral and 25% positive. Although the original corpus has only annotated one target per tweet, without specifying the location of the target, we expand this notion to consider cases where the target entity may appear more than once at different locations in the tweet, e.g.:

“Nicki Minaj has brought back the female rapper. - really? Nicki Minaj is the biggest parody in popular music since the Lonely Island.”

Semantically it is more appropriate and meaningful to consider both target appearances when determining the sentiment polarity of “Nicki Minaj” expressed in this tweet. While it isn’t clear if Dong et al. (2014) and Tang et al. (2016a) have considered this realistic **same-target-multi-appearance scenario**, Vo et al. (2015) and Zhang et al. (2016) do not take it into account when extracting target-dependent contexts. Contrary to these studies we extend our system to fully incorporate the situation where a target appears multiple times at different locations in the tweet. We add another pooling layer in (2) where we apply a *medium* pooling function to combine extracted feature vectors from each target appearance together into the final feature vector for the sentiment classification of such targets. Now the feature extraction function $P(X)$ in (2) becomes:

$$P(X) = [P_{medium}([f_1(X_1), \dots, f_1(X_m)]), \dots \dots], \quad (3)$$

$$P_{medium}([f_k(X_1), \dots, f_k(X_m)])]$$

where m is the number of appearances of the target and P_{medium} represents the dimension-wise *medium* pooling function.

Models: To investigate different ways of modelling target-specific context and evaluate the benefit of incorporating the same-target-multi-appearance scenario, we build these models:

- **Semeval-best:** is a tweet-level model using various types of features, namely ngrams, lexica and word embeddings with extensive data pre-processing and feature engineering. We use this model as a target-independent baseline as it approximates and beats the best performing system (Boag et al., 2015) in Semeval 2015 task 10. It also outperforms the highest ranking system, Tweester, on the Semeval 2016 corpus (by +4.0% in macro-averaged recall) and therefore constitutes a state-of-the-art tweet level baseline.
- **Naive-seg models:** **Naive-seg-** slices each tweet into a sequence of sub-sentences by using punctuation (i.e. ‘,’ ‘.’ ‘?’ ‘!’). Embedding features are extracted from each sub-sentence and pooling functions are applied to combine word vectors. **Naive-seg** extends it by adding features extracted from the left-target-right contexts, while **Naive-seg+** extends Naive-seg by adding lexicon filtered sentiment features.
- **TDParse models:** as described in Section 4.1. **TDParse-** uses a dependency parser to extract a syntactic parse tree to the target and map all child nodes to low-dimensional vectors. Final feature vectors for each target are generated using neural pooling functions. While **TDParse** extends it by adding features extracted from the left-target-right contexts, **TDParse+** uses three sentiment lexica for filtering words. **TDParse+ (m)** differs from **TDParse+** by taking into account the ‘same-target-multi-appearance’ scenario. Both **TDParse+** and **TDParse+ (m)** outperform state-of-the-art target-specific models.
- **TDPWindow-N:** the same as **TDParse+** with a window to constrain the left-right context.

For example if $N = 3$ then we only consider 3 tokens on each side of the target when extracting features from the left-right context.

4.2 Experimental Settings

To compare our proposed models with Vo & Zhang (2015), we have used the same pre-trained embedding resources and pooling functions (i.e. *max*, *min*, *mean*, *standard deviation* and *product*). For classification we have used LIBLINEAR (Fan et al., 2008), which approximates a linear SVM. In tuning the cost factor C we perform five-fold cross validation on the training data over the same set of parameter values for both Vo and Zhang (2015)’s implementation and our system. This makes sure our proposed models are comparable with those of Vo and Zhang (2015).

Evaluation metrics: We follow previous work on target-dependent Twitter sentiment classification, and report our performance in accuracy, 3-class macro-averaged (i.e. negative, neutral and positive) F_1 score as well as 2-class macro-averaged (i.e. negative and positive) F_1 score⁸, as used by the Semeval competitions (Rosenthal et al., 2015) for measuring Twitter sentiment classification performance.

4.3 Experimental results and comparison with other baselines

We report our experimental results in **Table 2** on the single-target benchmarking corpus (Dong et al., 2014), with three model categories: 1) tweet-level target-independent models, 2) target-dependent models without considering the ‘same-target-multi-appearance’ scenario and 3) target-dependent models incorporating the ‘same-target-multi-appearance’ scenario. We include the models presented in the previous section as well as models for target specific sentiment from the literature where possible.

Among the target-independent baseline models **Target-ind** (Vo and Zhang, 2015) and **Semeval-best** have shown strong performance compared with **SSWE** (Tang et al., 2014) and **SVM-ind** (Jiang et al., 2011) as they use more features, especially rich automatic features using the embeddings of Mikolov et al. (2013). Interestingly they also perform better than some of the target-dependent baseline systems, namely **SVM-dep**

⁸Note that this isn’t a binary classification task; the F_1 score is still effected by the neutral tweets.

(Jiang et al., 2011), **Recursive NN** and **AdaRNN** (Dong et al., 2014), showing the difficulty of fully extracting and incorporating target information in tweets. Basic **LSTM** models (Tang et al., 2016a) completely ignore such target information and as a result do not perform as well.

Among the target-dependent systems neural network baselines have shown varying results. The adaptive recursive neural network, namely **AdaRNN** (Dong et al., 2014), adaptively selects composition functions based on the input data and thus performs better than a standard recursive neural network model (**Recursive NN** (Dong et al., 2014)). **TD-LSTM** and **TC-LSTM** from Tang et al. (2016a) model left-target-right contexts using two LSTM neural networks and by doing so incorporate target-dependent information. **TD-LSTM** uses two LSTM neural networks for modeling the left and right contexts respectively. **TC-LSTM** differs from (and outperforms) **TD-LSTM** in that it concatenates target word vectors with embedding vectors of each context word. We also test the Gated recurrent neural network models proposed by Zhang et al. (2016) on the same dataset. The gated models include: **GRNN**, that includes gates in its recurrent hidden layers, **G3** that connects left-right context using a gated NN structure, and a combination of the two - **GRNN+G3**. Results show these gated neural network models do not achieve state-of-the-art performance. When we compare our target-dependent model **TDParse+**, which incorporates target-dependent features from syntactic parses, against the target-dependent models proposed by Vo and Zhang (2015), namely **Target-dep** which combines full tweet (pooled) word embedding features with features extracted from left-target-right contexts and **Target-dep+** that adds target-dependent sentiment features on top of **Target-dep**, we see that our method beats both of these, without using full tweet features⁹. **TDParse+** also outperforms the state-of-the-art **TC-LSTM**.

When considering the ‘same-target-multi-appearance’ scenario, our best model - **TDParse+ (m)** in Table 2). Even though **TDParse** doesn’t use lexica, it shows competitive results to **Target-dep+** which uses lexicon filtered sen-

⁹Note that the results reported in Vo and Zhang (2015) (71.1 in accuracy and 69.9 in F_1) were not possible to reproduce by running their code with very fine parameter tuning, as suggested by the authors

Model	Accuracy	3 Class F_1	2 Class F_1
SSWE	62.4	60.5	
SVM-ind	62.7	60.2	
LSTM	66.5	64.7	
Target-ind	67.05	63.4	58.5
Semeval-best	67.6	64.3	59.2
SVM-dep	63.4	63.3	
Recursive NN	63.0	62.8	
AdaRNN	66.3	65.9	
Target-dep	70.1	67.4	63.2
Target-dep+	70.5	68.1	64.1
TD-LSTM	70.8	69.0	
TC-LSTM	71.5	69.5	
GRNN	68.5	65.8	61.0
G3	68.5	67.0	63.9
GRNN+G3	67.9	65.2	60.5
TDParse+	72.1	69.8	66.0
Target-dep+ (m)	70.7	67.8	63.4
Naive-seg-	63.0	57.6	51.5
Naive-seg	70.8	68.4	64.5
Naive-seg+	70.7	67.7	63.2
TDParse-	61.7	57.0	51.1
TDParse	71.0	68.4	64.3
TDParse+ (m)	72.5	70.3	66.6
TDPWindow-2	68.2	64.7	59.2
TDPWindow-7	71.2	68.5	64.2
TDPWindow-12	70.5	67.9	63.8

Table 2: Performance comparison on the benchmarking data (Dong et al., 2014)

timent features. In the case of **TDParse-**, which uses exclusively features from syntactic parses, while it performs significantly worse than **Target-ind**, that uses only full tweet features, when the former is used in conjunction with features from left-target-right contexts it achieves better results than the equivalent **Target-dep** and **Target-dep+**. This indicates that syntactic target information derived from parses complements well with the left-target-right context representation. Clausal segmentation of tweets or sentences can provide a simple approximation to parse-tree based models (Li et al., 2015). In Table 2 we can see our naive tweet segmentation models **Naive-seg** and **Naive-seg+** also achieve competitive performance suggesting to some extent that such simple parse-tree approximation preserves the semantic structure of text and that useful target-specific information can be drawn from each segment or clause rather than the entire tweet.

5 Evaluating Baselines for target-specific sentiment in a multi-target setting

We perform multi-target-specific sentiment classification on our election dataset by extending

and applying our models described in Section 4.1. We compare the results with our other developed baseline models in Section 4.1, including a tweet-level model **Semeval-best** and clausal-segmentation models that provide simple parse-tree approximation, as well as state-of-the-art target-dependent models by Vo and Zhang (2015) and Zhang et al. (2016). The experimentation setup is the same as described in Section 4.2¹⁰.

Data set: Our election data has a training/testing ratio of 3.70, containing 3210 training tweets with 9912 target entities and 867 testing tweets with 2675 target entities.

Models: In order to limit our use of external resources we do not include **Naive-seg+** and **TD-Parser+** for evaluation as they both use lexica for feature generation. Since most of our tweets here contain $N > 1$ targets and the target-independent classifiers produce a single output per tweet, we evaluate its result N times against the ground truth labels, to make different models comparable.

Results: Overall the models perform much poorer than for the single-target benchmarking corpus, especially in 2-class F_1 score, indicating the challenge of the multi-target-specific sentiment recognition. As seen in Table 3 though the feature-rich tweet-level model **Semeval-best** gives a reasonably strong baseline performance (same as in Table 2), both it and **Target-ind** perform worse than the target-dependent baseline models **Target-dep/Target-dep+** (Vo and Zhang, 2015), indicating the need to capture and utilise target-dependent signals in the sentiment classification model. The Gated neural network models - **G3/GRNN/GRNN+G3** (Zhang et al., 2016) also perform worse than **Target-dep+** while the combined model - **GRNN+G3** fails to boost performance, presumably due to the small corpus size.

Our final model **TDParser** achieves the best performance especially in 3-class F_1 and 2-class F_1 scores in comparison with other target-dependent and target-independent models. This indicates that our proposed models can provide better and more balanced performance between precision and recall. It also shows the target-dependent syntactic information acquired from parse-trees is beneficial to determine the target’s sentiment particularly when used in conjunction with the left-

¹⁰Class weight parameter is not optimised for all experiments, though better performances can be achieved here by tuning the class weight due to the class imbalance nature of this dataset.

Model	Accuracy	3 Class F_1	2 Class F_1
Semeval-best	54.09	42.60	40.73
Target-ind	52.30	42.19	40.50
Target-dep	54.36	41.50	38.91
Target-dep+	55.85	43.40	40.85
GRNN	54.92	41.22	38.57
G3	55.70	41.40	37.87
GRNN+G3	54.58	41.04	39.46
Naive-seg-	51.89	39.94	37.17
Naive-seg	55.07	43.89	40.69
TDParser-	52.53	42.71	40.67
TDParser	56.45	46.09	43.43
TDPWindow-2	55.10	43.81	41.36
TDPWindow-7	55.70	44.66	41.35
TDPWindow-12	56.82	45.45	42.69

Table 3: Performance comparison on the election dataset¹¹

S1	Semeval-best	Target-dep+	TDParser
Macro 3-class- F_1	50.11	46.24	47.08
Micro 3-class- F_1	59.72	55.82	57.47
Macro 2-class- F_1	46.59	43.42	42.95
S2	Semeval-best	Target-dep+	TDParser
Macro 3-class- F_1	37.15	41.81	43.07
Micro 3-class- F_1	45.17	51.66	52.05
Macro 2-class- F_1	37.05	39.75	40.92
S3	Semeval-best	Target-dep+	TDParser
Macro 3-class- F_1	35.08	42.83	51.26
Micro 3-class- F_1	38.16	46.05	53.07
Macro 2-class- F_1	35.17	40.53	50.14

Table 4: Performance analysis in S1, S2 and S3

target-right contexts originally proposed by Vo and Zhang (2015) and in a scenario of multiple targets per tweet. Our clausal-segmentation baseline - **Naive-seg** models approximate such parse-trees by identifying segments of the tweet relevant to the target, and as a result **Naive-seg** achieves competitive performance compared to other baselines.

5.1 State-of-the-art tweet level sentiment vs target-specific sentiment in a multi-target setting

To fully compare our multi-target-specific models against other target-dependent and target-independent baseline methods, we conduct an additional experiment by dividing our election data test set into three disjoint subsets, on the basis of number of distinct target sentiment values per tweet: (**S1**) contains tweets having only one target sentiment, where the sentiment towards each target is the same; (**S2**) and (**S3**) contain two and three different types of targeted sentiment respec-

¹¹Any further results will be shared on our Github page: <https://goo.gl/S2T1GO>

tively (i.e. in **S3**, positive, neutral and negative sentiment are all expressed in each tweet). As described in Section 3.2, there are 2,051, 1,753 and 273 tweets in **S1**, **S2** and **S3** respectively.

Table 4 shows results achieved by the tweet-level target-independent model - **Semeval-best**, the state-of-the-art target-dependent baseline model - **Target-dep+**, and our proposed final model - **TDParse**, in each of the three subsets. We observe **Semeval-best** performs the best in **S1** compared to the two other models but its performance gets worse when different types of target sentiment are mentioned in the tweet. It has the worst performance in **S2** and **S3**, which again emphasises the need for multi-target-specific sentiment classification. Finally, our proposed final model **TDParse** achieves better performance than **Target-dep+** consistently over all subsets indicating its effectiveness even in the most difficult scenario **S3**.

6 Conclusion and Future work

In this work we introduce the challenging task of multi-target-specific sentiment classification for tweets. To help the study we have generated a multi-target Twitter corpus on UK elections which will be made publicly available. We develop a state-of-the-art approach which utilises the syntactic information from parse-tree in conjunction with the left-right context of the target. Our method outperforms previous approaches on a benchmarking single-target corpus as well as our new multi-target election data. Future work could investigate sentiment connections among all targets appearing in the same tweet as a multi-target learning task, as well as a hybrid approach that applies either Semeval-best or TDParse depending on the number of targets detected in the tweet.

Acknowledgments

We would like to thank Duy-Tin Vo, Meishan Zhang and Duyu Tang for sharing their implementation code respectively, which we have used for system performance comparison. We would also like to thank Li Dong for sharing their data, and City University London for recruiting PhD students for the annotation of our election corpus.

References

- Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.
- William Boag, Peter Potash, and Anna Rumshisky. 2015. Twitterhawk: A feature bucket based approach to sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 640–646, Denver, Colorado, June. Association for Computational Linguistics.
- Pete Burnap, Rachel Gibson, Luke Sloan, Rosalyn Southern, and Matthew Williams. 2016. 140 characters to victory?: Using twitter to predict the uk 2015 general election. *Electoral Studies*, 41:230–233.
- Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Baltimore, Maryland, June. Association for Computational Linguistics.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archana Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar, October. Association for Computational Linguistics.
- J Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.
- Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary

- for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal, September. Association for Computational Linguistics.
- Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 170–173, Barcelona, Spain, July. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California, June. Association for Computational Linguistics.
- Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).
- Shubham Pateria and Prafulla Choubey. 2016. AK-TSKI at semeval-2016 task 5: Aspect based sentiment analysis for consumer reviews. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 318–324.
- Nataliia Plotnikova, Micha Kohl, Kevin Volkert, Stefan Evert, Andreas Lerner, Natalie Dykes, and Heiko Ermer. 2015. Klueless: Polarity classification and association. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 619–625, Denver, Colorado, June. Association for Computational Linguistics.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado, June. Association for Computational Linguistics.
- Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11*, pages 1524–1534, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 451–463, Denver, Colorado, June. Association for Computational Linguistics.
- Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland, June. Association for Computational Linguistics.
- Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, Osaka, Japan, December. The COLING 2016 Organizing Committee.
- Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224, Austin, Texas, November. Association for Computational Linguistics.
- Richard Townsend, Adam Tsakalidis, Yiwei Zhou, Bo Wang, Maria Liakata, Arkaitz Zubiaga, Alexandra Cristea, and Rob Procter. 2015. Warwick-dcs: From phrase-based to target-specific sentiment recognition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 657–663, Denver, Colorado, June. Association for Computational Linguistics.
- Andranik Tumasjan, Timm Oliver Sprenger, Philipp G. Sandner, and Isabell M. Welp. May 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10:178–185.
- Saúl Vargas, Richard McCreddie, Craig Macdonald, and Iadh Ounis. 2016. Comparing overall and targeted sentiments in social media during crises. In *Proceedings of the Tenth International Conference on Web and Social Media, Cologne, Germany, May 17-20, 2016.*, pages 695–698.
- Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15*, pages 1347–1353. AAAI Press.
- Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120,

Jeju Island, Korea, July. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16*, pages 3087–3093. AAAI Press.

Annotating Derivations: A New Evaluation Strategy and Dataset for Algebra Word Problems

Shyam Upadhyay¹ and Ming-Wei Chang²

¹University of Illinois at Urbana-Champaign, IL, USA

²Microsoft Research, Redmond, WA, USA

upadhya3@illinois.edu

minchang@microsoft.com

Abstract

We propose a new evaluation for automatic solvers for algebra word problems, which can identify mistakes that existing evaluations overlook. Our proposal is to evaluate such solvers using *derivations*, which reflect how an equation system was constructed from the word problem. To accomplish this, we develop an algorithm for checking the equivalence between two derivations, and show how derivation annotations can be semi-automatically added to existing datasets. To make our experiments more comprehensive, we include the derivation annotation for DRAW-1K, a new dataset containing 1000 general algebra word problems. In our experiments, we found that the annotated derivations enable a more accurate evaluation of automatic solvers than previously used metrics. We release derivation annotations for over 2300 algebra word problems for future evaluations.

1 Introduction

Automatically solving math reasoning problems is a long-pursued goal of AI (Newell et al., 1959; Bobrow, 1964). Recent work (Kushman et al., 2014; Shi et al., 2015; Koncel-Kedziorski et al., 2015) has focused on developing solvers for *algebra word problems*, such as the one shown in Figure 1. Developing a solver for word problems can open several new avenues, especially for online education and intelligent tutoring systems (Kang et al., 2016). In addition, as solving word problems requires the ability to understand and analyze natural language, it serves as a good test-bed for evaluating progress towards goals of artificial intelligence (Clark and Etzioni, 2016).

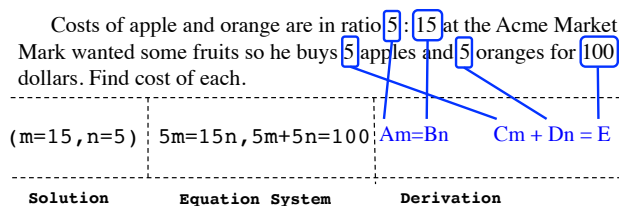


Figure 1: An algebra word problem with its *solution*, *equation system* and *derivation*. Evaluating solvers on derivation is more reliable than evaluating on solution or equation system, as it reveals errors that other metric overlook.

An automatic solver finds the solution of a given word problem by constructing a *derivation*, consisting of an un-grounded equation system¹ ($\{Am = Bn, Cm + Dn = E\}$ in Figure 1) and *alignments* of numbers in the text to its coefficients (blue edges). The derivation identifies a grounded equation system $\{5m = 15n, 5m + 5n = 100\}$, whose *solution* can then be generated to answer the problem. A derivation precisely describes *how* the grounded equation system was constructed from the word problem by the automatic solver. On the other hand, the grounded equation systems and the solutions are less informative, as they do not explain which span of text aligns to the coefficients in the equations.

While the derivation is clearly the most informative structure, surprisingly, no prior work evaluates automatic solvers using derivations directly. To the best of our knowledge, none of the current datasets contain human-annotated derivations, possibly due to the belief that the current evaluation metrics are sufficient and the benefit of evaluating on derivations is minor. Currently, the most popular evaluation strategy is to use *solution accuracy* (Kushman et al., 2014; Hosseini et al., 2014; Shi et al., 2015; Koncel-Kedziorski et

¹Also referred to as a *template*. We use these two terms interchangeably.

al., 2015; Zhou et al., 2015; Huang et al., 2016), which computes whether the solution was correct or not, as this is an easy-to-implement metric. Another evaluation strategy was proposed in (Kushman et al., 2014), which finds an approximate derivation from the gold equation system and uses it to compare against a predicted derivation. We follow (Kushman et al., 2014) and call this evaluation strategy the *equation accuracy*.²

In this work, we argue that evaluating solvers against human labeled derivation is important. Existing evaluation metrics, like solution accuracy are often quite generous — for example, an incorrect equation system, such as,

$$\{m + 5 = n + 15, \quad m + n = 15 + 5\}, \quad (1)$$

can generate the correct solution of the word problem in Figure 1. While equation accuracy appears to be a stricter metric than solution accuracy, our experiments show that the approximation can mislead evaluation, by assigning higher scores to an inferior solver. Indeed, a correct equation system, $(5m = 15n, 5m + 5n = 100)$, can be generated by using a wrong template, $Am = Bn, Am + An = C$, and aligning numbers in the text to coefficients incorrectly. We show that without knowing the correct derivation at evaluation time, a solver can be awarded for the wrong reasons.

The lack of annotated derivations for word problems and no clear definition for comparing derivations present technical difficulties in using derivation for evaluation. In this paper, we address these difficulties and for the first time propose to evaluate the solvers using *derivation accuracy*. To summarize, the contributions of this paper are:

- We point out that evaluating using derivations is more precise compared to existing metrics. Moreover, contrary to popular belief, there is a meaningful gap between the derivation accuracy and existing metrics, as it can discover crucial errors not captured previously.
- We formally define when two derivations are equivalent, and develop an algorithm that can determine the same. The algorithm is simple

²Note that an approximation of the derivation is necessary, as there is no annotated derivation. From the brief description in their paper and the code released by Kushman et al. (2014), we found that their implementation assumes that the first derivation that matches the equations and generates the correct solution is the correct reference derivation against which predicted derivations are then evaluated.

Word Problem	x	We are mixing a solution of 32% sodium and another solution of 12% sodium. How many liters of 32% and 12% solution will produce 50 liters of a 20% sodium solution?
Textual Numbers	$\mathcal{Q}(x)$	$\{32_1, 12_1, 32_2, 12_2, 50, 20\}$
Equation System	y	$32m + 12n = 20 * 50,$ $m + n = 50$
Solution		$m = 20, n = 30$
Template	T	$Am + Bn = C * D,$ $m + n = C$
Coefficients	$\mathcal{C}(T)$	A, B, C, D
Alignments	A	$\{32_1 \rightarrow A, 12_1 \rightarrow B,$ $50 \rightarrow C, 20 \rightarrow D\}$
EquivTNum		$\{[32_1, 32_2], [12_1, 12_2]\}$
Derivation	z	(T, A)

Table 1: The symbols we used in the paper. Our proposed annotations are shown in **bold**. Equivalent textual numbers, described in EquivTNum, are distinguished with subscripts.

to implement, and can accurately detect the equivalence even if two derivations have very different syntactic forms.

- We annotated over 2300 word algebra problems³ with detailed derivation annotations, providing high quality labeled semantic parses for evaluating word problems.

2 Evaluating Derivations

We describe our notation and revisit the notion of derivation introduced in (Kushman et al., 2014). We then formalize the notion of derivation equivalence and provide an algorithm to determine it.

Structure of Derivation The word problem in Table 1 shows our notation, where our proposed annotations are shown in **bold**. We denote a word problem by x and an *equation system* by y .

An un-grounded equation system (or *template*) T is a family of equation systems parameterized by a set of coefficients $\mathcal{C}(T) = \{c_i\}_{i=1}^k$, where each coefficient c_i aligns to a *textual number* (e.g., *four*) in the word problem. We also refer to the coefficients as *slots* of the template. We use (A, B, C, \dots) to represent coefficients and (m, n, \dots) to represent the unknown variables in the templates.

Let $\mathcal{Q}(x)$ be the set of all the textual numbers in the problem x , and $\mathcal{C}(T)$ be the coefficients to be determined in the template T . An *alignment* is a set of tuples $A = \{(q, c) \mid q \in \mathcal{Q}(x), c \in \mathcal{C}(T) \cup \{\epsilon\}\}$ aligning textual numbers to coefficient slots,

³available at <https://aka.ms/datadraw>

where a tuple (q, ϵ) indicates that the number q is not relevant to the final equation system.

Note that there may be multiple semantically equivalent textual numbers. e.g., in Figure 1, either of the 32 can be aligned to coefficient slot A in the template. These equivalent textual numbers are marked in the `EquivTNum` field in the annotation. If two textual numbers $q, q' \in \text{EquivTNum}$, then we can align a coefficient slot to either q or q' , and generate a equivalent alignment.

An alignment A and a template T together identify a *derivation* $z = (T, A)$ of an equation system. Note that there may be multiple valid derivations, using one of the equivalent alignments. We assume there exists a routine `Solve(y)` that find the solution of an equation system. We use a Gaussian elimination solver for our `Solve` routine. We use hand-written rules and the quantity normalizer in Stanford CoreNLP (Manning et al., 2014) to identify textual numbers.

Derivation Equivalence We define two derivations (T_1, A_1) and (T_2, A_2) to be equivalent *iff* the corresponding templates T_1, T_2 and alignments A_1, A_2 are equivalent.

Intuitively, two templates T_1, T_2 are equivalent if they can generate the same space of equation systems – i.e., for every assignment of values to slots of T_1 , there exists an assignment of values to slots of T_2 such that they generate the same equation systems. For instance, template (2) and (3) below are equivalent

$$m = A + Bn \quad m = C - n \quad (2)$$

$$m + n = A \quad m - Cn = B. \quad (3)$$

because after renaming (A, B, C) to (B, C, A) respectively in template (2), and algebraic manipulations, it is identical to template (3). We can see that any assignment of values to *corresponding* slots will result in the same equation system.

Similarly, two alignments A_1 and A_2 are equivalent if corresponding slots from each template align to the same textual number. For the above example, the alignment $\{1 \rightarrow A, 3 \rightarrow B, 4 \rightarrow C\}$ in template (2), and alignment $\{1 \rightarrow B, 3 \rightarrow C, 4 \rightarrow A\}$ in template (3) are equivalent. Note that the alignment $\{1 \rightarrow A, 3 \rightarrow B, 4 \rightarrow C\}$ for (2) is not equivalent to $\{1 \rightarrow A, 3 \rightarrow B, 4 \rightarrow C\}$ in (3), because it does not respect variable renaming. Our definition also allows two alignments to be

Algorithm 1 Evaluating Derivation

Input: Predicted (T_p, A_p) and gold (T_g, A_g) derivation

Output: 1 if predicted derivation is correct, 0 otherwise

```

1: if  $|\mathcal{C}(T_p)| \neq |\mathcal{C}(T_g)|$  then  $\triangleright$  different # of coeff. slots
2:   return 0
3: end if
4:  $\Gamma \leftarrow \text{TEMPLEQUIV}(T_p, T_g)$ 
5: if  $\Gamma = \emptyset$  then  $\triangleright$  not equivalent templates
6:   return 0
7: end if
8: if  $\text{ALIGNEQUIV}(\Gamma, A_p, A_g)$  then  $\triangleright$  Check alignments
9:   return 1
10: end if
11: return 0
12:


---


13: procedure  $\text{TEMPLEQUIV}(T_1, T_2)$ 
14:    $\triangleright$  Note that here  $|\mathcal{C}(T_1)| = |\mathcal{C}(T_2)|$  holds
15:    $\Gamma \leftarrow \emptyset$ 
16:   for each 1-to-1 mapping  $\gamma : \mathcal{C}(T_1) \rightarrow \mathcal{C}(T_2)$  do
17:     match  $\leftarrow$  True
18:     for  $t = 1 \dots R$  do  $\triangleright R$ : Rounds
19:       Generate random vector  $\mathbf{v}$ 
20:        $A_1 \leftarrow \{(\mathbf{v}_i \rightarrow c_i)\}, A_2 \leftarrow \{(\mathbf{v}_i \rightarrow \gamma(c_i))\}$ 
21:       if  $\text{Solve}(T_1, A_1) \neq \text{Solve}(T_2, A_2)$  then
22:         match  $\leftarrow$  False; break
23:       end if
24:     end for
25:     if match then  $\Gamma \leftarrow \Gamma \cup \{\gamma\}$ 
26:   end for
27:   return  $\Gamma$   $\triangleright \Gamma \neq \emptyset$  iff the templates are equivalent
28: end procedure
29:


---


30: procedure  $\text{ALIGNEQUIV}(\Gamma, A_1, A_2)$ 
31:   for mapping  $\gamma \in \Gamma$  do
32:     if following holds true,
33:
34:        $(q, c) \in A_1 \iff \{(q, \gamma(c)) \text{ or } (q', \gamma(c))\} \in A_2$ 
35:
36:       where  $(q', q) \in \text{EquivTNum}$ 
37:     then return 1
38:   end if
39: end for
40: return 0
41: end procedure

```

equivalent, if they use textual numbers in equivalent positions for corresponding slots (as described by `EquivTNum` field).

In the following, we carefully explain how template and alignment equivalence are determined algorithmically. Algorithm 1 shows the complete algorithm for comparing two derivations.

Template Equivalence We propose an approximate procedure `TEMPLEQUIV` (line 13) that detects equivalence between two templates. The procedure relies on the fact that under appropriate renaming of coefficients, two equivalent templates will generate equations which have the same solutions, for all possible coefficient assignments.

For two templates T_1 and T_2 , with the same number of coefficients $|\mathcal{C}(T_1)| = |\mathcal{C}(T_2)|$, we represent a choice of renaming coefficients by γ , a

1-to-1 mapping from $\mathcal{C}(T_1)$ to $\mathcal{C}(T_2)$. The two templates are equivalent if there exists a γ such that solutions of the equations identified by T_1 and T_2 are same, for all possible coefficient assignments. The `TEMPLEQUIV` procedure exhaustively tries all possible renaming of coefficients (line 16), checking if the solutions of the equation systems generated from a random assignment (line 19) match exactly. It declares equivalence if for a renaming γ , the solutions match for $R = 10$ such random assignments.⁴ The procedure returns all renamings Γ of coefficients between two templates under which they are equivalent (line 27). We discuss its effectiveness in §3.

Alignment Equivalence The `TEMPLEQUIV` procedure returns every mapping γ in Γ under which the templates were equivalent (line 4). Recall that γ identifies corresponding slots, c and $\gamma(c)$, in T_1 and T_2 respectively. We describe alignment equivalence using these mappings.

Two alignments A_1 and A_2 are equivalent if corresponding slots (according to γ) align to the same textual number. More formally, if we find a mapping γ such that for each tuple (q, c) in A_1 there is $(q, \gamma(c))$ in A_2 , then the alignments are equivalent (line 33). We allow for equivalent textual numbers (as identified by `EquivTNum` field) to match when comparing tuples in alignments.

The proof of correctness of Algorithm 1 is sketched in the appendix. Using Algorithm 1, we can define *derivation accuracy*, to be 1 if the predicted derivation (T_p, A_p) and the reference derivation (T_g, A_g) are equivalent, and 0 otherwise.

Properties of Derivation Accuracy By comparing derivations, we can ensure that the following errors are detected by the evaluation.

Firstly, correct solutions found using incorrect equations will be penalized, as the template used will not be equivalent to reference template. Secondly, correct equation system obtained by an incorrect template will also be penalized for the same reason. Lastly, if the solver uses the correct template to get the correct equation system, but aligns the wrong number to a slot, the alignment will not be equivalent to the reference alignment, and the solver will be penalized too.

⁴Note that this procedure is a Monte-Carlo algorithm, and can be made more precise by increasing R . We found making R larger than 10 did not have an impact on the empirical results.

We will see some illustrative examples of above errors in §5.3. Note that the currently popular evaluation metric of solution accuracy will not detect *any* of these error types.

3 Annotating Derivations

As none of the existing benchmarks contain derivation annotations, we decided to augment existing datasets with these annotations. We also annotated `DRAW-1K`, a new dataset of 1000 general algebra word problems to make our study more comprehensive. Below, we describe how we reduced annotation effort by semi-automatic generated some annotations.

Annotating gold derivations from scratch for all problems is time consuming. However, not all word problems require manual annotation – sometimes all numbers appearing in the equation system can be uniquely aligned to a textual number without ambiguity. For such problems, the annotations are generated automatically.⁵ We identify word problems which have at least one *alignment ambiguity* – multiple textual numbers with the same value, which appears in the equation system. An example of such a problem is shown in Figure 1, where there are three textual numbers with value 5, which appears in the equation system. Statistics for the number of word problems with such ambiguity is shown in Table 2.

We only ask annotators to resolve such alignment ambiguities, instead of annotating the entire derivation. If more than one alignments are genuinely correct (as in word problem of Table 1), we ask the annotators to mark both (using the `EquivTNum` field). This ensures our derivation annotations are *exhaustive* – all correct derivations are marked. With the correct alignment annotations, templates for all problems can be easily induced.

Annotation Effort To estimate the effort required to annotate derivations, we timed our annotators when annotating 50 word problems (all involved alignment ambiguities). As a control, we also asked annotators to annotate the entire derivation from scratch (i.e., only provided with the word problem and equations), instead of only fixing alignment ambiguities. When annotating from scratch, annotators took an average of 4 minute per word problem, while when fixing alignment ambiguities this time dropped to average of 1 minute

⁵Annotations for *all* problems are manually verified later.

Dataset	DRAW-1K	ALG-514	DOLPHIN-L
# problems	1000	514	832
w/ ambiguity vocab.	21%	23%	35%
	2.21k	1.83k	0.33k
Number of Templates			
before	329	30	273
after	224	24	203
% reduction	32%	20%	25%

Table 2: Statistics of the datasets. At least 20% of problems in each dataset had alignment ambiguities that required human annotations. The number of templates before and after annotation is also shown (reduction > 20%).

per word problem. We attained a inter-annotator agreement of 92% (raw percentage agreement), with most disagreements arising on EquivTNum field.⁶

Reconciling Equivalent Templates The number of templates has been used as a measure of dataset diversity (Shi et al., 2015; Huang et al., 2016), however prior work did not reconcile the equivalent templates in the dataset. Indeed, if two templates are equivalent, we can replace one with the other and still generate the correct equations. Therefore, after getting human judgements on alignments, we reconcile all the templates using TEMPLEQUIV as the final step of annotation.

TEMPLEQUIV is quite effective (despite being approximate), reducing the number of templates by at least 20% for all datasets (Table 2). We did not find any false positives generated by the TEMPLEQUIV in our manual examination. The reduction in Table 2 clearly indicates that equivalent templates are quite common in all datasets, and number of templates (and hence, dataset diversity) can be significantly overestimated without proper reconciliation.

4 Experimental Setup

We describe the three datasets used in our experiments. Statistics comparing the datasets is shown in Table 2. In total, our experiments involve over 2300 word problems.

Alg-514 The dataset ALG-514 was introduced in (Kushman et al., 2014). It consists of 514 general algebra word problems ranging over a variety of narrative scenarios (distance-speed, object counting, simple interest, etc.).

⁶These were adjudicated on by the first author.

Dolphin-L DOLPHIN-L is the linear-T2 subset of the DOLPHIN dataset (Shi et al., 2015), which focuses on *number word problems* – algebra word problems which describe mathematical relationships directly in the text. All word problems in the linear-T2 subset of the DOLPHIN dataset can be solved using linear equations.

DRAW-1K Diverse Algebra Word (DRAW-1K), consists of 1000 word problems crawled from algebra.com. Details on the dataset creation can be found in the appendix. As ALG-514 was also crawled from algebra.com, we ensured that there is little overlap between the datasets.

We randomly split DRAW-1K into train, development and test splits with 600, 200, 200 problems respectively. We use 5-fold cross validation splits provided by the authors for DOLPHIN-L and ALG-514.

4.1 Evaluation

We compare derivation accuracy against the following evaluation metrics.

Solution Accuracy We compute solution accuracy by checking if each number in the reference solution appears in the generated solution (disregarding order), following previous work (Kushman et al., 2014; Shi et al., 2015).

Equation Accuracy An approximation of derivation accuracy that is similar to the one used in Kushman et al. (2014). We approximate the reference derivation \tilde{z} by randomly chosen from the (several possible) derivations which lead to the gold y from x . Derivation accuracy is computed against this (possibly incorrect) reference derivation. Note that in equation accuracy, the approximation is used instead of annotated derivation. We include the metric of equation accuracy in our evaluations to show that human annotated derivation is necessary, as approximation made by equation accuracy might be problematic.

4.2 Our Solver

We train a solver using a simple modeling approach inspired by Kushman et al. (2014) and Zhou et al. (2015). The solver operates as follows. Given a word problem, the solver ranks all templates seen during training, Γ_{train} , and selects the set of the top- k (we use $k = 10$) templates $\Pi \subset \Gamma_{train}$. Next, all possible derivations $\mathcal{D}(\Pi)$ that use a template from Π are generated

Setting	Soln. Acc.	Eqn. Acc.	Deriv. Acc.
ALG-514			
TE	76.2	72.7	75.5
TD	78.4	73.9	77.8
TD - TE	2.2	1.2	2.3
DRAW-1K			
TE	52.0	48.0	48.0
TD	55.0	48.0	53.0
TD - TE	3.0	0	5.0
DOLPHIN			
TE	55.1	50.1	44.2
TD	57.5	36.8	54.9
TD - TE	2.4	-13.3	10.7

Table 3: TE and TD compared using different evaluation metrics. Note that while TD is clearly superior to TE due to extra supervision using the annotations, only derivation accuracy is able to correctly reflect the differences.

and scored. The equation system \hat{y} identified by highest scoring derivation \hat{z} is output as the prediction. Following (Zhou et al., 2015), we do not model the alignment of nouns phrases to variables, allowing for tractable inference when scoring the generated derivations. The solver is trained using a structured perceptron (Collins, 2002). We extract the following features for a (x, z) pair,

Template Features. Unigrams and bigrams of lemmas and POS tags from the word problem x , conjoined with $|\mathcal{Q}(x)|$ and $|\mathcal{C}(T)|$.

Alignment Tuple Features. For two alignment tuples, $(q_1, c_1), (q_2, c_2)$, we add features indicating whether c_1 and c_2 belong to the same equation in the template or share the same variable. If they belong to the same sentence, we also add lemmas of the nouns and verbs between q_1 and q_2 in x .

Solution Features. Features indicating if the solution of the system identified by the derivation are integer, negative, non-negative or fractional.

5 Experiments

Are solution and equation accuracy equally capable as derivation accuracy at distinguishing between good and bad models? To answer this question, we train the solver under two settings such that one of the settings has clear advantage over the other, and see if the evaluation metrics reflect this advantage. The two settings are,

Setting	Soln. Acc.	Eqn. Acc.	Deriv. Acc.
DRAW-1K + Alg-514			
TE	32.5	31.5	29.5
TE*	60.5	56.0	54.0
TD	62.0	53.0	59.5
TD - TE*	1.5	-3.0	5.5
DRAW-1K + Dolphin			
TE	41.0	37.5	37.5
TE*	58.5	55.5	51.5
TD	60.0	53.0	58.0
TD - TE*	1.5	-2.5	6.5

Table 4: When combining two datasets, it is essential to reconcile templates across datasets. Here TE* denotes training on equations after reconciling the templates, while TE simply combines datasets naively. As TE* represents a more appropriate setting, we compare TE* and TD in this experiment.

TE (TRAIN ON EQUATION) Only the (x, y) pairs are provided as supervision. Similar to (Kushman et al., 2014; Zhou et al., 2015), the solver finds a derivation which agrees with the equation system and the solution, and trains on it. Note that the derivation found by the solver may be incorrect.

TD (TRAIN ON DERIVATION) (x, z) pairs obtained by the derivation annotation are used as supervision. This setting trains the solver on human-labeled derivations. Clearly, the TD setting is a more informative supervision strategy than the TE setting. TD provides the correct template and correct alignment (i.e. labeled derivation) as supervision and is expected to perform better than TE, which only provides the question-equation pair.

We first present the main results comparing different evaluation metrics on solvers trained using the two settings.

5.1 Main Results

We compare the evaluation metrics in Table 3. We want to determine to what degree each evaluation metric reflects the superiority of TD over TE.

We note that solution accuracy always exceeds derivation accuracy, as a solver can sometimes get the right solutions even with the wrong derivation. Also, solution accuracy is not as sensitive as derivation accuracy to improvements in the solver. For instance, solution accuracy only changes by 2.4 on Dolphin-L when comparing TE and TD, whereas derivation accuracy changes

by 10.7 points. We found that the large gap on Dolphin-L was due to several alignment errors in the predicted derivations, which were detected by derivation accuracy. Recall that over 35% of the problems in Dolphin-L have alignment ambiguities (Table 2). In the TD setting, many of these errors made by our solver were corrected as the gold alignment was part of supervision.

Equation accuracy too has several limitations. For DRAW-1K, it cannot determine which solver is better and assigns them the same score. Furthermore, it often (incorrectly) considers TD to be a worse setting than TE, as evident from decrease in the scores (for instance, on DOLPHIN-L). Recall that equation accuracy attempts to approximate derivation accuracy by choosing a random derivation agreeing with the equations, which might be incorrect.

Study with Combining Datasets With several ongoing annotation efforts, it is a natural question to ask is whether we can leverage multiple datasets in training to generalize better. In Table 4, we combine DRAW-1K’s train split with other datasets, and test on DRAW-1K’s test split. DRAW-1K’s test split was chosen as it is the largest test split with general algebra problems (recall Dolphin-L contains only number word problems).

We found that in this setting, it was important to reconcile the templates *across* datasets. Indeed, when we simply combine the two datasets in the TE setting, we notice a sharp drop in performance (compared to Table 3). However, if we reconciled all templates and then used the new equations for training (called TE* setting in Table 4), we were able to see improvements from training on more data. We suspect difference in annotation style led to several equivalent templates in the combined dataset, which got resolved in TE*. Therefore, in Table 4, we compare TE* and TD settings.⁷

In Table 4, a trend similar to Table 3 can be observed – solution accuracy assigns a small improvement to TD over TE*. Derivation accuracy clearly reflects the fact that TD is superior to TE*, with a larger improvement compared to solution accuracy (eg., 5.5 vs 1.5). Equation accuracy, as before, considers TD to be worse than TE*.

Note that this experiment also shows that differences in annotation styles across different algebra problem datasets can lead to poor performance

⁷In TE*, the model still trains only using equations, without access to derivations. So TD is still better than TE*.

Dataset	Ours	KAZB	Best Result
ALG-514	76.2	68.7	79.7 (ZDC)
DOLPHIN-L	55.1	37.5	46.3 [‡] (SWLLR)
DRAW-1K	52.0	43.2	–

Table 5: Comparison of our solver and other state-of-the-art systems, when trained under TE setting. All numbers are solution accuracy. See footnote for details on the comparison to SWLLR.

when combining these datasets naively. Our findings suggest that derivation annotation and template reconciliation are crucial for such multi-data supervision scenarios.

5.2 Comparing Solvers

To ensure that the results in the previous section were not an artifact of any limitations of our solver, we show here that our solver is competitive to other state-of-the-art solvers, and therefore it is reasonable to assume that similar results can be obtained with other automatic solvers.

In Table 5, we compare our solver to KAZB, the system of Kushman et al. (2014), when trained under the existing supervision paradigm, TE (i.e., training on equations) and evaluated using solution accuracy. We also report the best scores on each dataset, using ZDC and SWLLR to denote the systems of Zhou et al. (2015) and Shi et al. (2015) respectively. Note that our system and KAZB are the only systems that can process all three datasets without significant modification, with our solver being clearly superior to KAZB.

5.3 Case Study

We discuss some interesting examples from the datasets, to show the limitations of existing metrics, which derivation accuracy overcomes.

Correct Solution, Incorrect Equation In the following example from the DOLPHIN-L dataset, by choosing the correct template and the wrong alignments, the solver arrived at the correct solutions, and gets rewarded by solution accuracy.

The sum of $2(q_1)$ numbers is $25(q_2)$. $12(q_3)$
less than $4(q_4)$ times one(q_5) of the numbers is
 $16(q_6)$ more than twice(q_7) the other number.
Find the numbers.

[‡]SWLLR also had a solver which achieves 68.0, using over 9000 semi-automatically generated rules tailored to number word problems. We compare to their similarity based solver instead, which does not use any such rules, given that the rule-based system cannot be applied to general word problems.

Note that there are seven textual numbers (q_1, \dots, q_7) in the word problem. We can arrive at the correct equations ($\{m + n = 25, 4m - 2n = 16 + 12\}$), by the correct derivation,

$$m + n = q_2 \quad q_4m - q_7n = q_6 + q_3.$$

However, the solver found the following derivation, which produces the incorrect equations ($\{m + n = 25, 2m - n = 2 + 12\}$),

$$m + n = q_2 \quad \mathbf{q_1}m - \mathbf{q_5}n = \mathbf{q_7} + q_3.$$

Both the equations have the same solutions ($m = 13, n = 12$), but the second derivation is clearly using incorrect reasoning.

Correct Equation, Incorrect Alignment In such cases, the solver gets the right equation system, but derived it using wrong alignment. Solution accuracy still rewards the solver. Consider the problem from the DOLPHIN-L dataset,

The larger of two(q_1) numbers is $2(q_2)$ more than $4(q_3)$ times the smaller. Their sum is $67(q_4)$. Find the numbers.

The correct derivation for this problem is,

$$m - q_3n = q_2 \quad m + n = q_4.$$

However, our system generated the following derivation, which although results in the exact same equation system (and thus same solutions), is clearly incorrect due incorrect choice of "two",

$$m - q_3n = \mathbf{q_1} \quad m + n = q_4.$$

Note that derivation accuracy will penalize the solver, as the alignment is not equivalent to the reference alignment (q_1 and q_2 are not semantically equivalent textual numbers).

Bad Approx. in Equation Accuracy The following word problem is from the ALG-514 dataset:

Mrs. Martin bought $3(q_1)$ cups of coffee and $2(q_2)$ bagels and spent $12.75(q_3)$ dollars. Mr. Martin bought $2(q_4)$ cups of coffee and $5(q_5)$ bagels and spent $14.00(q_6)$ dollars. Find the cost of one(q_7) cup of coffee and that of one(q_8) bagel.

The correct derivation is,

$$q_1m + q_2n = q_3 \quad q_4m + q_5n = q_6.$$

However, we found that equation accuracy used the following incorrect derivation for evaluation,

$$q_1m + \mathbf{q_2}n = q_3 \quad \mathbf{q_2}m + q_5n = q_6.$$

Note while this derivation does generate the correct equation system and solutions, the derivation utilizes the wrong numbers and misunderstood the word problem. This example demonstrates the needs to evaluate the quality of the word problem solvers using the annotated derivations.

6 Related Work

We discuss several aspects of previous work in the literature, and how it relates to our study.

Existing Solvers Current solvers for this task can be divided into two broad categories based on their inference approach – *template-first* and *bottom-up*. Template-first approaches like (Kushman et al., 2014; Zhou et al., 2015) infer the derivation $z = (T, A)$ *sequentially*. They first predict the template T and then predict alignments A from textual numbers to coefficients. In contrast, bottom-up approaches (Hosseini et al., 2014; Shi et al., 2015; Koncel-Kedziorski et al., 2015) *jointly* infer the derivation $z = (T, A)$. Inference proceeds by identifying parts of the template (eg. $Am + Bn$) and aligning numbers to it ($\{2 \rightarrow A, 3 \rightarrow B\}$). At any intermediate state during inference, we have a partial derivation, describing a fragment of the final equation system ($2m + 3n$). While our experiments used a solver employing the template-first approach, it is evident that performing inference in all such solvers requires constructing a derivation $z = (T, A)$. Therefore, annotated derivations will be useful for evaluating *all* such solvers, and may also aid in debugging errors.

Other reconciliation procedures are also discussed (though briefly) in earlier work. Kushman et al. (2014) reconciled templates by using a symbolic solver and removing pairs with the same canonicalized form. Zhou et al. (2015) also reconciled templates, but do not describe how it was performed. We showed that reconciliation is important for correct evaluation, for reporting dataset complexity, and also when combining multiple datasets.

Labeling Semantic Parses Similar to our work, efforts have been made to annotate semantic parses for other tasks, although primarily for providing supervision. Prior to the works of Liang et al. (2009) and Clarke et al. (2010), semantic parsers were trained using annotated logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007, *inter alia*), which were expensive to annotate. Recently, Yih et al. (2016) showed that labeled semantic parses for the knowledge based question answering task can be obtained at a cost comparable to obtaining answers. They showed significant improvements in performance of a question-answering system using the labeled parses instead of answers for training. More recently, by treating word problems as a semantic parsing task, Upadhyay et al. (2016) found that joint learning using both explicit (derivation as labeled semantic parses) and implicit supervision signals (solution as responses) can significantly outperform models trained using only one type of supervision signal.

Other Semantic Parsing Tasks We demonstrated that response-based evaluation, which is quite popular for most semantic parsing problems (Zelle and Mooney, 1996; Berant et al., 2013; Liang et al., 2011, *inter alia*) can overlook reasoning errors for algebra problems. A reason for this is that in algebra word problems there can be several semantic parses (i.e., derivations, both correct and incorrect) that can lead to the correct solution using the input (i.e., textual number in word problem). This is not the case for semantic parsing problems like knowledge based question answering, as correct semantic parse can often be identified given the question and the answer. For instance, paths in the knowledge base (KB), that connect the answer and the entities in the question can be interpreted as legitimate semantic parses. The KB therefore acts as a constraint which helps prune out possible semantic parses, given only the problem and the answer. However, such KB-based constraints are unavailable for algebra word problems.

7 Conclusion and Discussion

We proposed an algorithm for evaluating derivations for word problems. We also showed how derivation annotations can be easily obtained by only involving annotators for ambiguous cases. We augmented several existing benchmarks with

derivation annotations to facilitate future comparisons. Our experiments with multiple datasets also provided insights into the right approach to combine datasets – a natural step in future work. Our main finding indicates that derivation accuracy leads to a more accurate assessment of algebra word problem solvers, finding errors which other metrics overlook. While we should strive to build such solvers using as little supervision as possible for training, having high quality annotated data is essential for correct evaluation.

The value of such annotations for evaluation becomes more immediate for online education scenarios, where such word solvers are likely to be used. Indeed, in these cases, merely arriving at the correct solution, by using incorrect reasoning may prove detrimental for teaching purposes. We believe derivation based evaluation closely mirrors how humans are evaluated in schools (by forcing solvers to show “their work”).

Our datasets with the derivation annotations have applications beyond accurate evaluation. For instance, certain solvers, like the one in (Roy and Roth, 2015), train a *relevance classifier* to identify which textual numbers are relevant to solving the word problem. As we only annotate relevant numbers in our annotations, our datasets can provide high quality supervision for such classifiers. The datasets can also be used in evaluation test-beds, like the one proposed in (Koncel-Kedziorski et al., 2016).

We hope our datasets will open new possibilities for the community to simulate new ideas and applications for automatic problem solvers.

Acknowledgments

The first author was supported on a grant sponsored by DARPA under agreement number FA8750-13-2-0008. We would also like to thank Subhro Roy, Stephen Mayhew and Christos Christodoulopoulos for useful discussions and comments on earlier versions of the paper.

References

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

- Daniel G. Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 591–614. ACM.
- Peter Clark and Oren Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37(1):5–12.
- James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world’s response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27, Uppsala, Sweden, July. Association for Computational Linguistics.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.
- Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar, October. Association for Computational Linguistics.
- Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896, Berlin, Germany, August. Association for Computational Linguistics.
- Bo Kang, Arun Kulshreshtha, and Joseph J. LaViola Jr. 2016. Analyticalink: An interactive learning environment for math word problem solving. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 419–430. ACM.
- Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.
- Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California, June. Association for Computational Linguistics.
- Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland, June. Association for Computational Linguistics.
- Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore, August. Association for Computational Linguistics.
- Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
- Allen Newell, John C. Shaw, and Herbert A. Simon. 1959. Report on a general problem-solving program. In *IFIP Congress*, volume 256, page 64.
- Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal, September. Association for Computational Linguistics.
- Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, Lisbon, Portugal, September. Association for Computational Linguistics.
- Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from Explicit and Implicit Supervision Jointly For Algebra Word Problems. In *Proceedings of EMNLP*, pages 297–306.
- Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.
- Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 201–206, Berlin, Germany, August. Association for Computational Linguistics.

J. M. Zelle and R. J. Mooney. 1996. Learning to Parse Database Queries using Inductive Logic Programming. In *AAAI*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822, Lisbon, Portugal, September. Association for Computational Linguistics.

A Creating DRAW-1K

We crawl over 100k problems from <http://algebra.com>. The 100k word problems include some problems which require solving non-linear equations (e.g. finding roots of quadratic equations). We filter out these problems using keyword matching. We also filter problems whose explanation do not contain a variable named “x”. This leaves us with 12k word problems.

Extracting Equations A word problem on algebra.com is accompanied by a detailed explanation provided by instructors. In our crawler, we use simple pattern matching rules to extract all the equations in the explanation. The problems often have sentences which are irrelevant to solving the word problem (e.g. “Please help me, I am stuck.”). During cleaning, the annotator removes such sentences from the final word problem and performs some minor editing if necessary.⁸

1000 problems were randomly chosen from these pool of 12k problems, which were then shown to annotators as described earlier to get the derivation annotations.

B Proof of Correctness (Sketch)

For simplicity, we will assume that `EquivTNum` is empty. The proof can easily be extended to handle the more general situation.

⁸In some cases, some of the numbers in the text are rephrased (“10ml” to “10 ml”) in order to allow NLP pipeline work properly.

Lemma 1. The procedure `TEMPLEQUIV` returns $\Gamma \neq \emptyset$ iff templates T_1, T_2 are equivalent (w.h.p.).

Proof First we prove that with high probability we are correct in claiming that a γ found by the algorithm leads to equivalence. Let probability of getting the same solution even when the template are not equivalent be $\epsilon(T_1, T_2, \gamma) < 1$. The probability that solution is same for R rounds for T_1, T_2 which are not equivalent is $\leq \epsilon^R$, which can be made arbitrarily small by choosing large R . Therefore, with a large enough R , obtaining $\Gamma \neq \emptyset$ from `TEMPLEQUIV` implies there is a γ under which templates generate equations with the same solution, and by definition, are equivalent.

Conversely, if templates are equivalent, it implies $\exists \gamma^*$ such that under that mapping for any assignment, the generated equations have the same solution. As we iterate over all possible 1-1 mappings γ between the two templates, we will find γ^* eventually.

Proposition Algorithm 1 returning 1 implies derivations (T_p, A_p) and (T_g, A_g) are equivalent.

Proof Algorithm returns 1 only if `TEMPLEQUIV` found a $\Gamma \neq \emptyset$, and $\exists \gamma \in \Gamma$, following holds

$$(q, c) \in A_g \iff (q, \gamma(c)) \in A_p$$

i.e., the corresponding slots aligned to the same textual number. `TEMPLEQUIV` found a $\Gamma \neq \emptyset$ implies templates are equivalent (w.h.p). Therefore, $\exists \gamma \in \Gamma$ such that the corresponding slots aligned to the same textual number implies the alignments are equivalent under mapping γ . Together they imply that the derivation was equivalent (w.h.p.).

An Extensive Empirical Evaluation of Character-Based Morphological Tagging for 14 Languages

Georg Heigold

DFKI & Saarland University
Saarbrücken, Germany
georg.heigold@dfki.de

Günter Neumann

DFKI
Saarbrücken, Germany
neumann@dfki.de

Josef van Genabith

DFKI & Saarland University
Saarbrücken, Germany
josef.van.genabith@dfki.de

Abstract

This paper investigates neural character-based morphological tagging for languages with complex morphology and large tag sets. Character-based approaches are attractive as they can handle rarely and unseen words gracefully. We evaluate on 14 languages and observe consistent gains over a state-of-the-art morphological tagger across all languages except for English and French, where we match the state-of-the-art. We compare two architectures for computing character-based word vectors using recurrent (RNN) and convolutional (CNN) nets. We show that the CNN based approach performs slightly worse and less consistently than the RNN based approach. Small but systematic gains are observed when combining the two architectures by ensembling.

1 Introduction

Character-based approaches have been studied for many applications in natural language processing, including part-of-speech (POS) tagging (dos Santos and Zadrozny, 2014; Ling et al., 2015; Gillick et al., 2016; Plank et al., 2016; Ma and Hovy, 2016), morphological tagging (Labeau et al., 2015), parsing (Ballesteros et al., 2015), named entity recognition (Gillick et al., 2016), language modeling (Ling et al., 2015; Kim et al., 2016), and neural machine translation (Costajussà and Fonollosa, 2016). Character-based representations have the advantage of gracefully handling rare or unseen words and tend to produce more compact models as the number of atomic units, i.e., characters, is smaller compared to the number of words in word-level approaches. The issue of rare or unseen words is particularly pro-

nounced when working on morphologically-rich languages, small amounts of training data or noisy user input.

Morphological tagging is the task of assigning a morphological analysis to a token in context. The morphological analysis for a word consists of a sequence of feature:value pairs describing, for example, case, gender, person and tense. A particular concatenation of such feature:value pairs is referred to as a single tag (Oflazer and İlker Kuroz, 1994; Hajic and Hladka, 1998; Mueller et al., 2013).

Following (Müller and Schuetze, 2015), we also add the part-of-speech to this morphological tag and refer to it as POS-MORPH:

```
I see four words
                |
                POS=noun:CASE=acc:...
                ...:NUMBER=plural
```

Given a word in context, we predict a POS-MORPH tag as a complete unit, rather than as the individual component parts. This approach allows us to share large parts of the model but can only produce POS-MORPH analyses attested in the training data (cf. Table 2). This is still the standard approach to morphological tagging and disambiguation as, given sufficient amounts of training data, the number of POS-MORPH descriptions that cannot be produced usually is small.

Character-based POS tagging (rather than full POS-MORPH tagging) has been extensively evaluated in the literature (dos Santos and Zadrozny, 2014; Ling et al., 2015; Gillick et al., 2016; Plank et al., 2016). The results are competitive but do not systematically outperform the state of the art. Only Plank et al. (2016) report consistent gains by using shallow neural network architectures in combination with multitask learning, multilingual

learning, and pre-trained word embeddings.

State-of-the-art results for morphological tagging (full POS-MORPH tagging) can be found in (Mueller et al., 2013; Müller and Schuetze, 2015). To the best of our knowledge, there has not been much research on character-based morphological tagging so far. Labeau et al. (2015) is an exception but report results for German only. Their best results are on a par with state-of-the-art results. Heigold et al. (2016) show clear gains of character-based over state-of-the-art morphological taggers. However, the evaluation is limited to German and Czech.

Research on character-based approaches in general NLP clearly divides into papers that use CNN-based architectures (dos Santos and Zadrozny, 2014; Kim et al., 2016; Costa-jussà and Fonollosa, 2016) and papers that use LSTM-based architectures (Labeau et al., 2015; Ling et al., 2015; Gillick et al., 2016; Ballesteros et al., 2015; Plank et al., 2016; Ma and Hovy, 2016). There are a number of examples where an LSTM paper reports results of a CNN paper for comparison, such as (Ling et al., 2015) (POS tagging for English) and (Gillick et al., 2016) (named entity recognition for English). However, there is no direct comparison between CNN and LSTM based architectures in morphological tagging.

In this paper, we investigate character-based morphological tagging in more depth. More specifically, the contributions of this paper include:

- the evaluation of character-based morphological tagging on 14 different languages of different morphological complexity;
- the empirical comparison of long-short term memory (LSTM) and convolutional neural network (CNN) based architectures;
- the demonstration of systematic gains of our character-based, language-agnostic morphological tagger over a state-of-the-art morphological tagger across morphologically rich languages; moreover, and perhaps as expected, we show that the relative gains are clearly correlated with the amount of the training data;
- the evaluation of the complementarity of LSTM- and CNN-based architectures by ensemble experiments.

The remainder of the paper is organized as follows. Section 2 summarizes the character-based neural network approaches used in this paper. The data sets and model configurations are described in Section 3 and in Section 4, respectively. The empirical evaluation is presented in Section 5. Section 6 concludes the paper. The Appendix contains a listing of all experimental results obtained in this paper.

2 Character-based Tagging

We assume an input sentence w_1^N with (complex POS-MORPH morphological) output tags t_1^N and a zeroth-order Markov model

$$p(t_1^N | w_1^N) = \prod_{n=1}^N p(t_n | w_1^N) \quad (1)$$

whose factors are modeled by a suitable neural network. For character-based tagging, we use the character representation of the word, $w = c_1^M$. This assumes that the segmentation of the sentence into words is known, which is straightforward for the languages under consideration.

At the top level, each input word maps to one complex POS-MORPH morphological output tag. Hence, we can model the position-wise probabilities $p(t | w_1^N)$ with recurrent neural networks, such as long short-term memory recurrent neural networks (LSTMs) (Graves, 2012). Fig. 1 (a) shows such a network architecture where the inputs are the word vectors v_1^N . At the lower level, we use a CNN-based (Fig. 1 (b)) or an LSTM-based (Fig. 1 (c)) architecture to compute the character-based word vectors. As we are using bidirectional LSTMs (BLSTMs) at the top level, we shall refer to the complete architectures as CNNHighway-BLSTM and LSTM-BLSTM. The two architectures are fairly similar. In our opinion, however, there is an important difference between the two. CNNHighway is more constructive in the sense that it explicitly specifies the possible character context widths with a hard upper bound and defines an embedding size for each context width. LSTMs are more generic as they are claimed to implicitly learn these details (Schmidhuber, 1992).

The weights of the network, θ , are jointly estimated using conditional log-likelihood

$$F(\theta) = - \sum_{n=1}^N \log p_{\theta}(t_n | w_1^N). \quad (2)$$

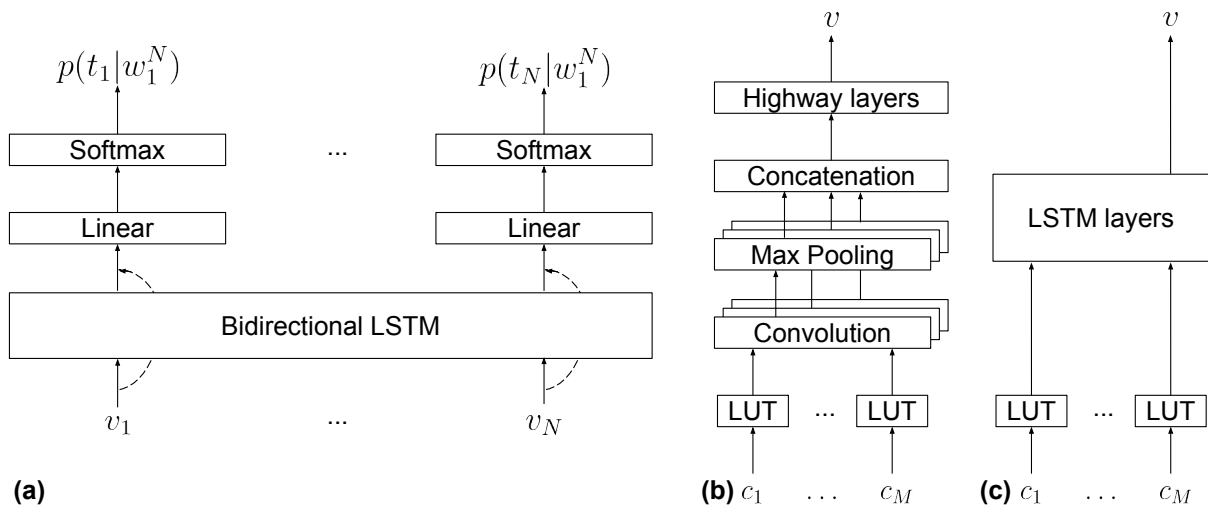


Figure 1: Character-based neural tagging architecture: (a) sub-network mapping word vectors v_1^N to tags t_1^N , dashed arrows indicate optional skip connections, (b) CNNs with different filter widths followed by fully-connected layers with highway connections (CNNHighway), and (c) deep LSTM using the last output to map the character string to a word vector. The networks in (b) and (c) are cloned to produce the input word vectors v_1^N in (a). LUT stands for lookup table.

Learning in recurrent or very deep neural networks is non-trivial and skip/shortcut connections have been proposed to improve the learning of such networks (Pascanu et al., 2014; He et al., 2016). We use such connections (dashed arrows in Fig. 1) for LSTM-BLSTM to alleviate potential learning issues.

At test time, the predicted tag sequence is the tag sequence that maximizes the conditional probability $p(t_1^N|w_1^N)$. For the factorization in Eq. (1), the search can be done position-wise. This significantly reduces the computational and implementation complexity compared to first-order Markov models as used in (Collobert et al., 2011; dos Santos and Zadrozny, 2014; Labeau et al., 2015).

3 Data

Most of the data sets are taken from the UD treebanks¹. We also use a number of older data sets in order to compare our results with existing results in the literature, including Czech/PDT², German/TIGER³, and Korean/SPMRL⁴. The corpus statistics for the different languages can be found in Table 1. The chosen languages are from different language families: Balto-Slavic (Bulgar-

ian, Czech, Russian), Finnic (Estonian, Finnish), Finno-Ugric (Hungarian), Germanic (German), Indo-Iranian (Hindi), Koreanic (Korean), Romance (Romanian, Semitic (Arabic), and Turkic (Turkish). They include several examples for both agglutinative and fusional languages. The amount of training data ranges from 33k training tokens (Hungarian/UD) to 1,174k training tokens (Czech/UD).

Table 2 summarizes the tag statistics for the different languages. The number of tags is the number of POS-MORPH tags occurring in the training data. We give the test entropy based on a unigram tag model estimated on the training data as a simple measure for the difficulty of the associated sequence classification problem. The type/token ratio (TTR), also known as vocabulary size divided by text length, is computed on 1M words from randomly selected sentences from a different data set⁵ and is a simple measure to quantify the morphological complexity of a language (Bane, 2008). A higher TTR value indicates higher morphological complexity.

¹<http://dependencies.org/>

²<https://ufal.mff.cuni.cz/pdt3.0>

³<http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html>

⁴<http://dokufarm.phil.hhu.de/spmrl2013/?animal=spmrl2013>

⁵The sentences are all taken from the Wiki dumps on <http://linguatools.org/tools/corpora/wikipedia-monolingual-corpora/> and <https://archive.org/details/wikipediadumps?&sort=-downloads&page=2>.

Table 1: Corpus statistics, $OOV_{\geq 5}$ denotes the percentage of test word tokens with five or more occurrences in the training data

Language	Train sentences (k)	Train tokens (k)	Test tokens (k)	$OOV_{\geq 5}$ (%)
Arabic/UD	6	256	32	20.7
Bulgarian/UD	9	124	16	27.3
Czech/PDT	39	691	93	17.5
UD	68	1174	174	15.7
English/UD	13	205	25	16.7
Estonian/UD	15	188	24	32.2
Finnish/UD	12	163	9	38.9
French/UD	15	367	7	12.7
German/TIGER	40	760	92	17.2
Hindi/UD	13	281	35	10.1
Hungarian/UD	1	33	4	48.0
Korean/SPMRL	23	296	28	42.7
Romanian/UD	5	109	18	27.6
Russian/UD	47	815	108	19.5
Turkish/UD	4	42	9	46.5

Table 2: Tag statistics, TTR stands for Type/Token Ratio

Language	#Tags	Entropy	TTR (%)
Arabic/UD	320	32.5	12
Bulgarian/UD	448	49.5	12
Czech/PDT	878	77.7	11
UD	1418	97.7	11
English/UD	119	27.9	7
Estonian/UD	787	57.3	13
Finnish/UD	1593	76.1	17
French/UD	197	34.1	8
German/TIGER	681	97.7	13
Hindi/UD	922	56.9	7
Hungarian/UD	652	64.5	14
Korean/SPMRL	1976	119.4	20
Romanian/UD	444	65.8	7
Russian/UD	434	54.6	16
Turkish/UD	987	73.0	10

4 Setups

We use the same model setups for LSTM-BLSTM and CNNHighway-BLSTM as in (Heigold et al., 2016). The hyper-parameters are set to

- CNNHighway: the large setup from (Kim et al., 2016), i.e., character vector size = 15, filter widths ranging from one to seven, number of filters as a function of the filter width $\min\{200, 50 \cdot \text{filter width}\}$, two highway layers
- LSTM: character vector size = 128, two layers with 1024 and 256 nodes

The BLSTM modeling the context of words in a sentence (Fig. 1 (a)) consists of two hidden layers, each with 256 hidden nodes.

These hyper-parameters were tuned on the German TIGER development data and are optimal on a "best effort basis." German is good for the hyper-parameter tuning as it is a relatively hard task (see Table 2) and shows morphological effects both within and across words. Furthermore, the TIGER corpus is relatively large, which reduces statistical fluctuations in training and testing. Apart from these considerations, the choice was random. Furthermore, we tested language-specific tuning for a few languages, but it does not seem to give further gains. Moreover, the network hyper-parameters were tuned to give best accuracy rather than most compact models or even comparable numbers of

parameters as our application is not constrained by memory or runtime. The hyper-parameters were then used for all languages. We ran the external tools MarMoT⁶ and JNN⁷ (see Appendix) with the suggested default values.

The networks are optimized as described in (Heigold et al., 2016). In particular, the optimization is done with RMSProp (Tieleman and Hinton, 2012), with a fixed initial learning rate and a learning rate decay of two every tenth epoch for German, TIGER, and is adjusted for the other languages according to the amount of training data. The batch size is always 16. Furthermore, we use dropout. The dropout probability is empirically set to 0.4 for Hungarian and Turkish, which only have a very limited amount of training data (Table 1), and to 0.2 for all other languages.

5 Empirical Evaluation

We empirically evaluate an LSTM-based and a CNN-based architecture for character-based morphological tagging (Section 2) and compare them against MarMoT, a state-of-the-art morphological tagger (Mueller et al., 2013). For the evaluation we use twelve different morphologically-rich languages with different characteristics, plus two morphologically-poor languages for contrastive results (Section 3). The configurations are described in Section 4.

Fig. 2 plots the relative gain over MarMoT (see Appendix A for more details) against the amount of training data. The horizontal dotted line at 0% indicates the MarMoT baseline. The blue squares are for LSTM-BLSTM results. Connecting them for the morphologically-rich languages shows a clear, nearly-linear dependency of the relative gain on the amount of training data. Only the data point for Turkish at 40% is an outlier (should be around 20%). This result suggests that compared to MarMoT, LSTM-BLSTM is very data efficient. Even for very small amounts of training data (e.g., 33k tokens for Hungarian), the relative gain is still 15%. On the other hand, more data helps. In case of Czech, increasing the amount of training data from 691k (Czech/PDT) to 1174k (Czech/UD) tokens leads to some additional gain and yields almost a 50% relative gain. It should be noted, however, that the two data sets use different tag sets, with the Czech/UD one being more complex than

the Czech/PDT (Table 2).

We use an LSTM-BLSTM of the same size for all languages, although the amount of training data varies by roughly two orders of magnitude. Therefore, it is a valid question if a larger model specifically designed for Czech/UD or a smaller model for Turkish/UD would improve the results. We have developed locally tuned and tested larger and smaller models in terms of number of nodes or layers but with similar or worse performance: -0.1% with more nodes (Czech) or approx. -1% with fewer nodes or fewer layers (Turkish). This observation suggests that the configuration optimized for German is fairly robust across many different languages, which is an attractive property from a practical perspective.

In contrast, we do not observe a gain of LSTM-BLSTM over MarMoT for English and French. Both languages are considered to be morphologically poor, as supported by the tag statistics in Table 2. This may be because of the low morphological complexity, i.e., a character representation does not add much information to a word representation. Another explanation might be that the linguistic experts have focused on English and French in the last decades and found a good set of features, which however does not well generalize to other, morphologically more complex languages.

It is tempting to analyze these results in more detail by splitting languages into sub-categories. Here, we refrain from doing so as it is delicate to draw conclusions from very small sample sizes (3-4 languages, say).

The green circles (in Fig. 2) are for CNNHighway-BLSTM results, a neural network architecture that has been developed for character-based language modeling (Kim et al., 2016). Overall, LSTM-BLSTM and CNNHighway-BLSTM perform similarly, see Fig. 2. Looking at the details, however, CNNHighway-BLSTM tends to perform slightly worse and less consistently than LSTM-BLSTM.

While LSTM-BLSTM and CNNHighway-BLSTM perform similarly they may capture complementary effects. To measure the complementarity of the two architectures, we build an ensemble consisting of the LSTM-BLSTM and the CNNHighway-BLSTM by taking the geometric mean of the scores. The accuracies are shown in Fig. 2 as LSTM+CNNHighway-BLSTM. Ex-

⁶<http://cistern.cis.lmu.de/marmot/>

⁷<https://github.com/wlin12/JNN>

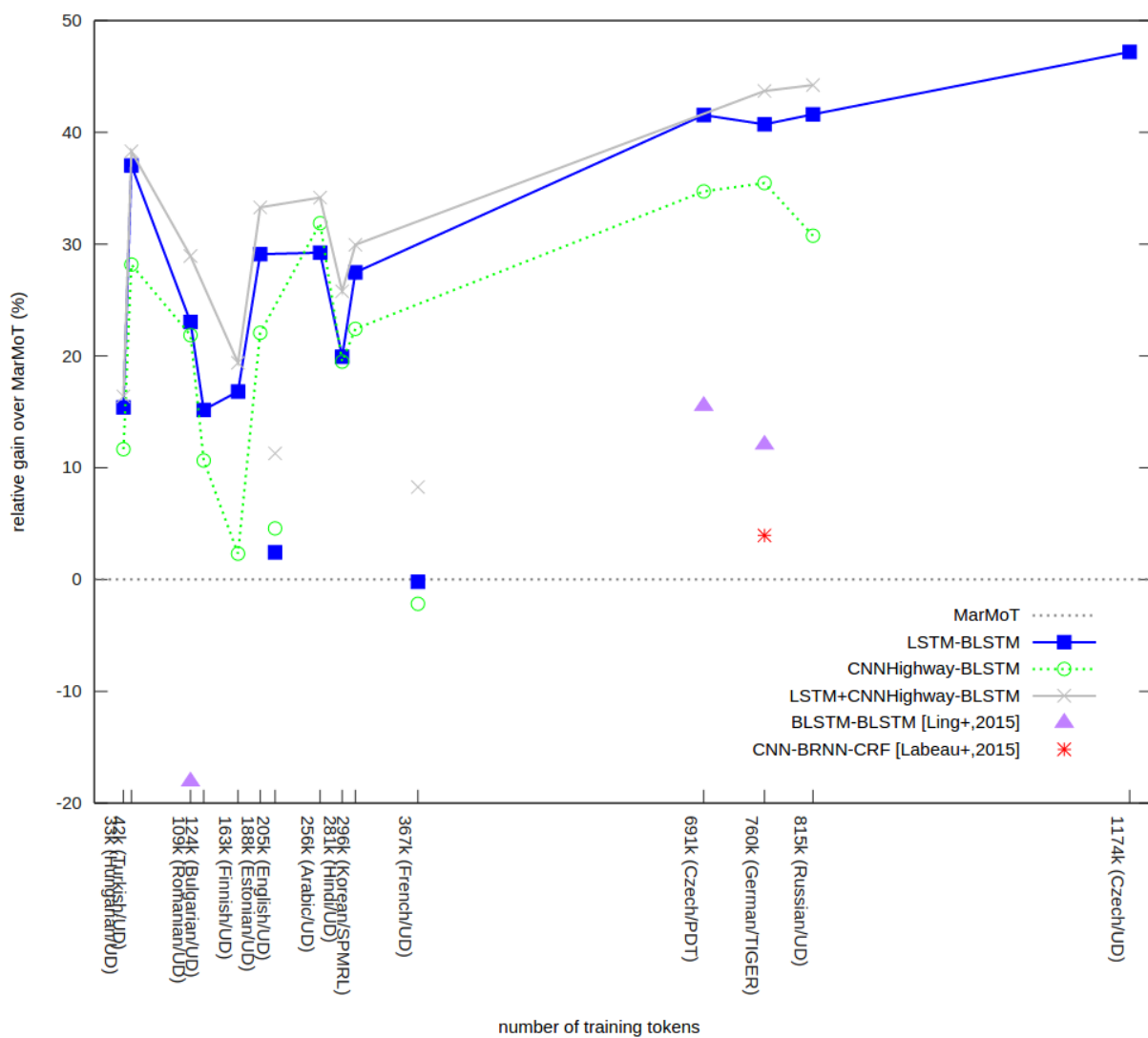


Figure 2: Relative gains (%) over MarMoT

cept maybe for English and French, we observe marginal but consistent gains over LSTM-BLSTM or CNNHhighway-BLSTM.

For additional comparison, we add a few additional points in the plot. The red cross indicates the result from (Labeau et al., 2015), which is a combination of a CNN, a bidirectional RNN, and a Markov model. The purple triangles are generated with the external tool JNN⁸, which implements a shallow BLSTM-BLSTM (i.e., only one bidirectional LSTM layer in each BLSTM). One might expect that this model performs better on smaller data sets. But actually, it is clearly worse both for large (Czech/PDT and German/TIGER) and small data sets (Romanian/UD).

6 Summary & Future Work

In this paper, we demonstrated that a character-based neural approach can achieve consistent improvements over a state-of-the-art morphological tagger (MarMoT). The evaluation included a dozen of languages of different morphological complexity and with different characteristics. The relative gains for the morphologically-rich languages range from 15% to almost 50%, with a clear dependency on the amount of training data. Several aspects are remarkable about this result.

First, these results use the same model architecture with the same number of layers and nodes, without any language-specific modifications. Further local language and training data setting specific tuning does not seem to help much.

Second, the neural approach seems to be more data efficient than the baseline tagger with manually designed features, also when only 30k training tokens are available.

Third, a fairly generic deep and hierarchical recurrent neural network architecture seems to perform as well or better than a more specialized convolutional neural network based architecture.

Fourth, to keep the setup as simple as possible, we have not used advanced techniques which are reported to lead to improvements, including a non-trivial structured prediction model (e.g., a first-order Markov model) (Collobert et al., 2011; dos Santos and Zadrozny, 2014; Labeau et al., 2015; Ma and Hovy, 2016), additional unsupervised data (e.g., via word2vec) (Müller and Schuetze, 2015; Ling et al., 2015; Plank et al., 2016; Ma and Hovy, 2016), combination of dif-

ferent word representations (Labeau et al., 2015; Ma and Hovy, 2016; Plank et al., 2016), multilingual learning (Gillick et al., 2016; Plank et al., 2016), and auxiliary tasks (Plank et al., 2016). Future work will include the investigation of these more advanced techniques. From this perspective, our paper provides a baseline for future research in multilingual character-based neural morphological tagging.

Last but not least, we do not observe any gains for English and French (except when using ensembles). This may be due to the low morphological complexity of these languages or because manual feature engineering has focused on these languages over the last decades with good results.

Acknowledgment

This work has been partly funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.
- Max Bane. 2008. Quantifying and measuring morphological complexity. In C.B. Chang and H.J. Haynie, editors, *Proceedings of the 26th West Coast Conference on Formal Linguistics*, pages 69–76. Cascadilla Proceedings Project, Somerville, MA, USA.
- Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany, August. Association for Computational Linguistics.
- Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amar-nag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California, June. Association for Computational Linguistics.

⁸<https://github.com/wlin12/JNN>

- Alex Graves. 2012. *Supervised sequence labelling with recurrent neural networks*. Studies in Computational Intelligence. Springer, Heidelberg, New York.
- Jan Hajic and Barbora Hladka. 1998. Tagging inflective languages: Prediction of morphological categories for a rich structured tagset. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 483–490, Montreal, Quebec, Canada, August. Association for Computational Linguistics.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.
- Georg Heigold, Günter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *CoRR*, abs/1606.06640.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *AAAI*, Phoenix, AZ, USA, February.
- Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237, Lisbon, Portugal, September. Association for Computational Linguistics.
- Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fernandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September. Association for Computational Linguistics.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August. Association for Computational Linguistics.
- Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Thomas Müller and Hinrich Schuetze. 2015. Robust morphological tagging with word representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536, Denver, Colorado, May–June. Association for Computational Linguistics.
- Kemal Oflazer and İlker Kuroz. 1994. Tagging and morphological disambiguation of Turkish text. In *Proceedings of the Applied natural language processing*.
- Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *ICLR*.
- Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany, August. Association for Computational Linguistics.
- Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, Beijing, China, June.
- Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.
- Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

A Raw Results

This appendix contains Table 3 with the raw results used in this paper. When available, the best comparable error rates from the literature are used. Otherwise, we produced the error rates with the publicly available tools and the suggested default values. More specifically, we used the state-of-the-art tagger MarMoT⁹ for the baselines and the LSTM-based POS tagger JNN¹⁰ for some contrastive results.

⁹<http://cistern.cis.lmu.de/marmot/>

¹⁰<https://github.com/wlin12/JNN>

Table 3: Tag error rates (%) on test sets, some of which are taken from the literature: (a) (Mueller et al., 2013), (b) (Labeau et al., 2015)

Language	MarMoT ⁹	CNN -biRNN-CRF	BLSTM -BLSTM ¹⁰	LSTM -BLSTM	CNNHighway -BLSTM
Arabic/UD	9.13			6.46	6.22
Bulgarian/UD	5.73			4.86	5.12
Czech/PDT	7.46 ^a		6.30	4.36	4.87
UD	6.97			3.68	
English/UD	7.00			6.83	6.68
Estonian/UD	8.11			5.75	6.32
Finnish/UD	7.79			6.48	7.61
French/UD	5.08			5.09	5.19
German/TIGER	11.42 ^a	10.97 ^b	10.04	6.77	7.37
Hindi/UD	11.44			9.16	9.21
Hungarian/UD	26.49			22.41	23.40
Korean/SPMRL	18.60			13.49	14.43
Romanian/UD	7.64		9.02	5.88	5.97
Russian/UD	6.08			3.55	4.21
Turkish/UD	17.28			10.88	12.41

Neural Multi-Source Morphological Reinflection

Katharina Kann

CIS
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell

Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze

CIS
LMU Munich, Germany

Abstract

We explore the task of multi-source morphological reinflection, which generalizes the standard, single-source version. The input consists of (i) a target tag and (ii) multiple pairs of source form and source tag for a lemma. The motivation is that it is beneficial to have access to more than one source form since different source forms can provide complementary information, e.g., different stems. We further present a novel extension to the encoder-decoder recurrent neural architecture, consisting of multiple encoders, to better solve the task. We show that our new architecture outperforms single-source reinflection models and publish our dataset for multi-source morphological reinflection to facilitate future research.

1 Introduction

Morphologically rich languages still constitute a challenge for natural language processing (NLP). The increased data sparsity caused by highly inflected word forms in certain languages causes otherwise state-of-the-art systems to perform worse in standard tasks, e.g., parsing (Ballesteros et al., 2015) and machine translation (Bojar et al., 2016). To create systems whose performance is not deterred by complex morphology, the development of NLP tools for the generation and analysis of morphological forms is crucial. Indeed, these considerations have motivated a great deal of recent work on the topic (Ahlberg et al., 2015; Dreyer, 2011; Nicolai et al., 2015).

In the area of generation, the most natural task is morphological inflection—finding an inflected form for a given target tag and lemma. An example for English is as follows: ($\text{trg}:3\text{rdSgPres}$, *bring*)

	Present Ind		Past Ind		Past Sbj	
	Sg	Pl	Sg	Pl	Sg	Pl
1	treffe	treffen	traf	trafen	träfe	träfen
2	triffst	trefft	trafst	traft	träfest	träfet
3	trifft	treffen	traf	trafen	träfe	träfen

Table 1: The paradigm of the strong German verb TREFFEN, which exhibits an irregular ablaut pattern. Different parts of the paradigm make use of one of four bolded theme vowels: **e**, **i**, **a** or **ä**. In a sense, the verbal paradigm is partitioned into subparadigms. To see why multi-source models could help in this case, starting only from the infinitive **treffen** makes it difficult to predict subjunctive form **träfest**, but the additional information of the fellow subjunctive form **träfe** makes the task easier.

\mapsto *brings*. In this case, the 3rd person singular present tense of *bring* is generated. One generalization of inflection is morphological reinflection (MRI) (Cotterell et al., 2016a), where we must produce an inflected form from a triple of target tag, source form and source tag. The inflection task is the special case where the source form is the lemma. As an example, we may again consider generating the English past tense form from the 3rd person singular present: ($\text{trg}:3\text{rdSgPres}$, *brought*, $\text{src}:\text{Past}$) \mapsto *brings* (where trg = “target tag” and src = “source tag”). As the starting point varies, MRI is more difficult than morphological inflection and exhibits more data sparsity. However, it is also more widely applicable since lexical resources are not always complete and, thus, the lemma is not always available. A more complex German example is given in Table 1.

In this work, we generalize the MRI task to a multi-source setup. Instead of using a single source form-tag pair, we use *multiple* source form-tag pairs. Our motivation is that (i) it is often beneficial to have access to more than one source form since different source forms can provide complementary information, e.g., different stems; and (ii)

in many application scenarios, we will have encountered more than one form of a paradigm at the point when we want to generate a new form.

We will make the intuition that multiple source forms provide complementary information precise in the next section, but first return to the English verb *bring*. Generating the form *brings* from *brought* may be tricky—there is an irregular vowel shift. However, if we had a second form with the same theme vowel, e.g., *bringing*, the task would be much easier, i.e., (`trg:3rdSgPres`, `form1:brought`, `src1:Past`, `form2:bringing`, `src2:Gerund`). A multi-source approach clearly is advantageous for this case since mapping *bringing* to *brings* is regular even though the verb itself is irregular.

The contributions of the paper are as follows. (i) We define the task of multi-source MRI, a generalization of single-source MRI. (ii) We show that a multi-source MRI system, implemented as a novel encoder-decoder, outperforms the top-performing system in the SIGMORPHON 2016 Shared Task on Morphological Reinflection on seven out of eight languages, when given additional source forms. (iii) We release our data to support the development of new systems for MRI.

2 The Task: Multi-Source Reinflection

Previous work on morphological reinflection has assumed a single source form, i.e., an input consisting of exactly one inflected source form (potentially the lemma) and the corresponding morphological tag. The output is generated from this input. In contrast, multi-source morphological reinflection, the task we introduce, is a generalization in which the model receives multiple form-tag pairs. In effect, this gives the model a partially annotated paradigm from which it predicts the rest.

The multi-source variant is a more natural problem than single-source morphological reinflection since we often have access to more than just one form.¹ For example, corpora such as the universal dependency corpus (McDonald et al., 2013) that are annotated on the token level with inflectional features often contain several different inflected forms of a lemma. Such corpora would provide an ideal source of data for the multi-source MRI task.

¹Scenarios where a single form is available and that form is the lemma are perhaps not infrequent. In high-resource languages, an electronic dictionary may have near-complete coverage of the lemmata of the language. However, paradigm completion is especially crucial for neologisms and low-resource languages.

Formally, we can think of a morphological paradigm as follows. Let Σ be a discrete alphabet for a given language and \mathcal{T} be the set of morphological tags in the language. The inflectional table or morphological paradigm π of a lemma w can be formalized as a set of pairs:

$$\pi(w) = \{(f_1, t_1), (f_2, t_2), \dots, (f_N, t_N)\}, \quad (1)$$

where $f_i \in \Sigma^+$ is an inflected form of w , and $t_i \in \mathcal{T}$ is the morphological tag of the form f_i . The integer N is the number of slots in the paradigm that have the syntactic category (POS) of w .

Using this notation, single-source morphological reinflection (MRI) can be described as follows. Given a target tag and a pair of source form and source tag ($t_{\text{trg}}, (f_{\text{src}}, t_{\text{src}})$) as input, predict the target form f_{trg} . There has been a substantial amount of prior work on this task, including systems that participated in Task 2 of the SIGMORPHON 2016 shared task (Cotterell et al., 2016a). Thus, we may define the task of *multi-source morphological reinflection* as follows: Given a target tag and a set of k form-tag source pairs ($t_{\text{trg}}, \{(f_{\text{src}}^1, t_{\text{src}}^1), \dots, (f_{\text{src}}^k, t_{\text{src}}^k)\}$) as input, predict the target form f_{trg} . Note that single-source MRI is a special case of multi-source MRI for $k = 1$.

2.1 Motivating Examples

Figure 1 gives examples for four different configurations that can occur in multi-source MRI.² We have colored the source forms green and drawn a dotted line to the target if they contain sufficient information for correct generation. If two source forms together are needed, the dotted line encloses both of them. Source forms that provide no information in the configuration are colored red (no arrow); note these forms could provide (and in most cases will provide) useful information for other combinations of source and target forms.

²Figure 1 is not intended as a complete taxonomy of possible MRI configurations, e.g., there are hybrids of ANYFORM and NOFORM (some forms are informative, others are suppletive) and fuzzy variants (a single form gives pretty good evidence for how to generate the target form, but another single form gives better evidence). All of our examples make additional assumptions, e.g., that we have not seen other similar forms in training either of the same lemma (e.g., *poner*) or of a similar lemma (e.g., *reponer*). Hopefully, the examples are illustrative of the main conceptual distinction: several single forms each are sufficient by themselves (ANYFORM), a single, but carefully selected form is sufficient (SINGLEFORM), multiple forms are needed to generate the target (MULTIFORM) and the target form cannot be predicted (irregular) from the source forms (NOFORM).

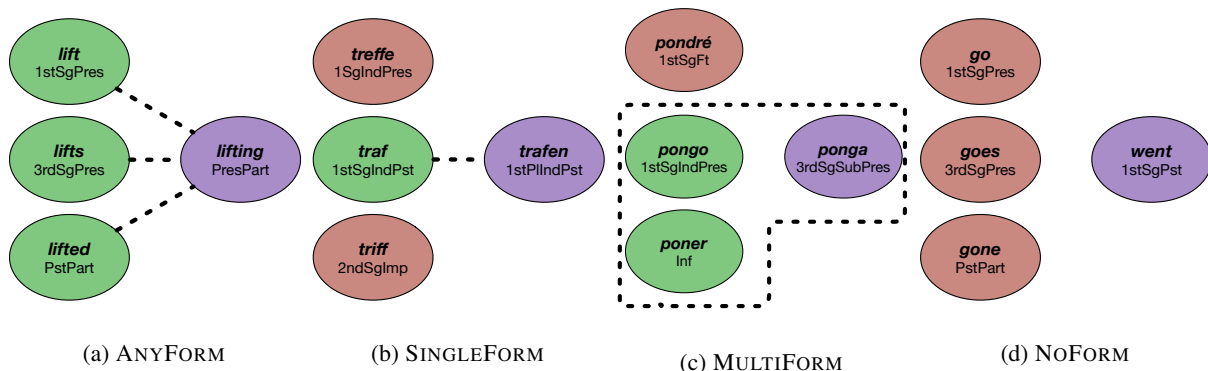


Figure 1: Four possible input configurations in multi-source morphological reinfection (MRI). In each subfigure, the target form on the right is **purple**. The source forms are on the left and are **green** if they can be used to predict the target form (also connected with a dotted line) and **red** if they cannot. There are four possible configurations: (i) ANYFORM is the case where one can predict the target form from any of the source forms. (ii) SINGLEFORM is the case where only one form can be used to regularly predict the target form. (iii) MULTIFORM is the case where multiple forms are *necessary* to predict the target form. (iv) NOFORM is the case where the target form cannot be regularly derived from any of the source forms. Multi-source MRI is expected to perform better than single-source MRI for the configurations SINGLEFORM and MULTIFORM, but not for the configurations ANYFORM and NOFORM.

The first type of configuration is ANYFORM: each of the available source forms in the subset of the English paradigm (*lift*, *lifts*, *lifted*) contains enough information for a correct generation of the target form *lifting*. The second configuration is SINGLEFORM: there is a single form that contains enough information for correct generation, but it has to be carefully selected. Inflected forms of the German verb *treffen* ‘to meet’ have different stem vowels (see Table 1). In single-source reinfection, producing a target form with one stem vowel (*a* in *trafe* in the figure) from a source form with another stem vowel (e.g., *e* in *treffe*) is difficult.³

In contrast, the learning problem for the SINGLEFORM configuration is much easier in multi-source MRI. The multi-source model does not have to learn the possible vowel changes of this irregular verb; instead, it just needs to pick the correct vowel change from the alternatives offered in the input. This is a relatively easy task since the theme vowel is identical. So we only need to learn one general fact about German morphology (which suffix to add) and will then be able to produce the correct form with high accuracy. This type of regularity is typical of complex morphology: there are groups of forms in a paradigm that are similar and it is highly predictable which of these groups a particular target form for a new word will be a member of. As long as one representative of each group is part of the multi-source input, we can select it to generate the correct form.

³It is not impossible to learn, but *treffen* is an irregular verb, so we cannot easily leverage the morphology we have learned about other verbs.

In the MULTISOURCE configuration, we are able to use information from multiple forms if no single form is sufficient by itself. For example, to generate *ponga*, 3rdSgSubPres of *poner* ‘to put’ in Spanish, we need to know what the stem is (*ponga*, not *pona*) and which conjugation class (*-ir*, *-er* or *-ar*) it is part of (*ponga*, not *pongue*). The single-source input *pongo*, 1stSgIndPres, does not reveal the conjugation class: it is compatible with both *ponga* and *pongue*. The single-source input *poner*, Inf, does not reveal the stem for the subjunctive: it is compatible with both *ponga* and *pona*—we need both source forms to generate the correct form *ponga*.

Again, such configurations are frequent cross-linguistically, either in this “discrete” variant or in more fuzzy variants where taking several forms together increases our chances of producing the correct target form. Finally, we call configurations NOFORM if the target form is completely irregular and not related to any of the source forms. The suppletive form *went* is our example for this case.

2.2 Principle Parts

The intuition behind the MRI task draws inspiration from the theoretical linguistic notion of **principle parts** (Finkel and Stump, 2007; Stump and Finkel, 2013). The notion is that a paradigm has a subset that allows for maximum predictability. In terms of language pedagogy, the principle parts would be a minimal set of forms a student has to learn in order to be able to generate any form in the paradigm. For instance for the partial German paradigm in Table 1, the forms *treffen*, *trifft*, *trafen*, and *träfen*

could form *one* potential set of principle parts.

From a computational learning point of view, maximizing predictability is always a boon—we want to make it as easy as possible for the system to learn the morphological regularities and subregularities of the language. Giving the system the principle parts as input is one way to achieve this.

3 Model Description

Our model is a multi-source extension of MED, Kann and Schütze (2016b)’s encoder-decoder network for MRI. In MED, a single bidirectional recurrent neural network (RNN) encodes the input. In contrast, we use multiple encoders to be able to handle multiple source form-tag pairs. In MED, a decoder RNN produces the output from the hidden representation. We do not change this part of the architecture, so there is still a single decoder.⁴

3.1 Input and Output Format

For k source forms, our model takes k different inputs of parallel structure. Each of the $1 \leq i \leq k$ inputs consists of the target tag t_{trg} and the source form f_i and its corresponding source tag t_i . The output is the target form. Each source form is represented as a sequence of characters; each character is represented as an embedding. Each tag—both the target tag and the source tags—is represented as a sequence of subtags; each subtag is represented as an embedding.

More formally, we define the alphabet Σ_{lang} as the set of characters in the language and Σ_{subtag} as the set of subtags that occur as part of the set of morphological tags \mathcal{T} of the language, e.g., if $1\text{st-SgPres} \in \mathcal{T}$, then 1st , Sg and $\text{Pres} \in \Sigma_{\text{subtag}}$. Each of the k inputs to our system is of the following format: $S_{\text{start}} \Sigma_{\text{subtag}}^+ \Sigma_{\text{lang}}^+ \Sigma_{\text{subtag}}^+ S_{\text{end}}$ where the first subtag sequence is the source tag t_i and the second subtag sequence is the target tag. The output format is: $S_{\text{start}} \Sigma_{\text{lang}}^+ S_{\text{end}}$, where the symbols S_{start} and S_{end} are predefined start and end symbols.

3.2 Multi-Source Encoder-Decoder

The encoder-decoder is based on the machine translation model of Bahdanau et al. (2015) and all specifics of our model are identical to the original presentation unless stated otherwise.⁵ Whereas

⁴The edit tree (Chrupała, 2008; Müller et al., 2015) augmentation discussed in Kann and Schütze (2016b) was not employed here.

⁵We modify the implementation of the model freely available at <https://github.com/mila-udem>.

Bahdanau et al. (2015)’s model has only one encoder, our model consists of $k \geq 1$ encoders and processes k sources simultaneously. The k sources have the form $X_m = (t_{trg}, f_{src}^m, t_{src}^m)$, represented as $S_{\text{start}} \Sigma_{\text{subtag}}^+ \Sigma_{\text{lang}}^+ \Sigma_{\text{subtag}}^+ S_{\text{end}}$ as described above. Characters and subtags are embedded.

The input to encoder m is X_m . Each encoder consists of a bidirectional RNN that computes a hidden state h_{mi} for each position, the concatenation of forward and backward hidden states. Decoding proceeds as follows:

$$p(y | X_1, \dots, X_k) = \prod_{t=1}^{|Y|} p(y_t | \{y_1, \dots, y_{t-1}\}, c_t) \\ = \prod_{t=1}^{|Y|} g(y_{t-1}, s_t, c_t), \quad (2)$$

where $y = (y_1, \dots, y_{|Y|})$ is the output sequence (a sequence of $|Y|$ characters), g is a nonlinear function, s_t is the hidden state of the decoder and c_t is the sum of the encoder states h_{mi} , weighted by attention weights $\alpha_{mi}(s_{t-1})$ that depend on the decoder state:

$$c_t = \sum_{m=1}^k \sum_{i=1}^{|X_m|} \alpha_{mi}(s_{t-1}) h_{mi}. \quad (3)$$

A visual depiction of this model may be found in Figure 2. A more complex hierarchical attention structure would be an alternative, but this simple model in which all hidden states contribute on the same level in a single attention layer (i.e., $\sum_{m=1}^k \sum_{i=1}^{|X_m|} \alpha_{mi} = 1$) works well as our experiments show. The k encoders share their weights.

4 Multi-Source Reinflection Experiment

We evaluate the performance of our model in an experiment based on Task 2 of the SIGMORPHON Shared Task on Morphological Reinflection (Cotterell et al., 2016a). This is a single-source MRI task as outlined in Section 1.

4.1 Experimental Settings

Datasets. Our datasets are based on the data from the SIGMORPHON 2016 Shared Task on Morphological Reinflection (Cotterell et al., 2016a). Our experiments cover eight languages: Arabic, Finnish, Georgian, German, Hungarian, Russian, Spanish and Turkish. The languages were chosen to represent different types of morphology. Finnish,

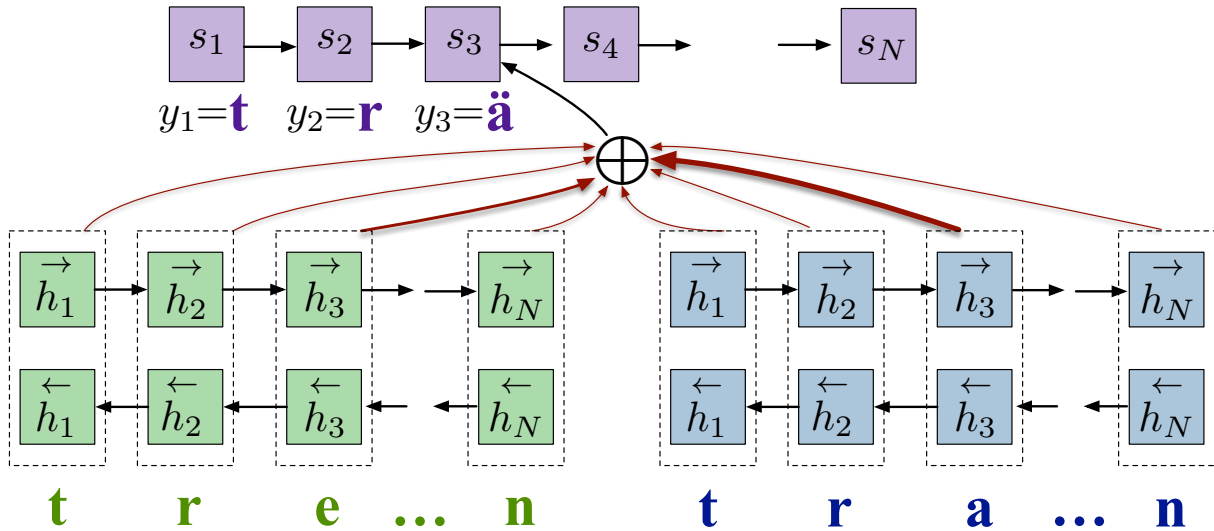


Figure 2: Visual depiction of our multi-source encoder-decoder RNN. We sketch a two encoder model, where the left encoder reads in the present form **treffen** and the right encoder reads in the past tense form **trafen**. They work together to predict the subjunctive form **träfen**. The shadowed red arcs indicate the strength of the attention weights—we see the network is focusing more on **a** because it helps the decoder better predict **ä** than **e**. We omit the source and target tags as input for conciseness.

German, Hungarian, Russian, Turkish and Spanish are all suffixing. In addition to being suffixing, three of these languages employ vocalic (German, Spanish) and consonantal (Russian) stem changes for many inflections. The members of the remaining sub-group are agglutinative. Georgian makes use of prefixation as well as suffixation. Arabic morphology contains both concatenative and templatic elements. We build multi-source versions of the dataset for Task 2 of the SIGMORPHON shared task in the following way. We use data from the UNIMORPH project,⁶ containing complete paradigms for all languages of the shared task. The shared task data was sampled from the same set of paradigms; our new dataset is a superset of the SIGMORPHON data.

We create our new dataset by uniformly sampling three additional word forms from the paradigm of each source form in the original data. In combination with the source and target forms of the original dataset, this means that our dataset is a set of 5-tuples consisting each of four source forms and one target form.⁷ Ideally, we would like to keep the experimental variable k , the number of sources we use in multi-source MRI, con-

⁶<http://unimorph.org>

⁷One thing to note is that the original shared task data was sampled depending on word frequency in unlabeled corpora. We do not impose a similar condition, so the frequency distributions of our data and the shared task data are different. Also, we excluded Maltese and Navajo due to a lack of data to create the additional multi-source datasets.

	1	2	3	≥ 4
ar	0	0	0	12,800
fi	0	0	0	12,800
ka	1015	84	2	11,699
de	0	0	0	12,800
hu	0	0	0	19,200
ru	0	0	5	12,794
es	1575	25	877	10,323
tu	0	0	0	12,800

Table 2: Number of target forms in the training set for which 1, 2, 3 or ≥ 4 source forms (in the training set) are available for prediction. The tables for the development and test splits show the same pattern and are omitted.

stant for a particular experiment or vary it systematically across other experimental conditions. Table 2 gives an overview of the number of different source forms per language in our dataset. Our dataset is available for download at <http://cistern.cis.lmu.de>.

Hyperparameters. We use embeddings of size 300. Our encoder and decoder GRUs have 100 hidden units each. Following Le et al. (2015), we initialize all encoder and decoder weights as well as the embeddings with an identity matrix. All biases are initialized with zero. We use stochastic gradient descent, Adadelata (Zeiler, 2012) and a minibatch size of 20 for training. Training is done for a maximum number of 90 epochs. If no improvement occurs for 20 epochs, we stop training early. The final model we run on test is the model

	source form(s) used					
	1	2	3	4	1–2	1–4
ar	.871	.813	.796	.830	.905	.944
fi	.956	.929	.941	.934	.965	.978
ka	.967	.943	.942	.934	.969	.979
de	.954	.922	.931	.912	.959	.980
hu	.992	.962	.963	.963	.988	.989
ru	.876	.795	.824	.817	.888	.911
es	.975	.961	.963	.968	.977	.984
tu	.967	.928	.947	.944	.970	.983

Table 3: Accuracy on MRI for single-source (1, 2, 3, 4) and multi-source (1–2, 1–4) models. Best result in bold.

that performs best on the development data.

Baselines. For the single-source case, we apply MED, the top-scoring system in the SIGMORPHON 2016 Shared Task on Morphological Reinflexion (Cotterell et al., 2016a; Kann and Schütze, 2016b). At the time of writing, MED constitutes the state of the art on the dataset. For Arabic, German and Turkish, we run an additional set of experiments to test two additional architectural configurations of multi-source encoder-decoders: (i) In addition to the default configuration in which all encoders share parameters, we also test the option of each encoder learning its own set of parameters (shared par’s: yes vs. no in Table 4). (ii) Another way of realizing a multi-source system is to concatenate all sources and give this to an encoder-decoder with a single encoder as one input (encoders: $k = 1$ vs. $k > 1$ in Table 4).

Evaluation Metric. We evaluate on 1-best accuracy (exact match) against the gold form. We deviate from the shared task, which also evaluates under mean reciprocal rank and edit distance. We omit the later two since all these metrics were highly correlated (Cotterell et al., 2016a).

4.2 Results

Table 3 shows the results of the MRI experiment on test data. We compare using a single source, the first two sources and all four sources. The first source (in column “1”) is the original source from the SIGMORPHON shared task. Recall that we used uniform sampling to identify additional forms whereas the sampling procedure of the shared task took into account frequency. We suspect that this is the reason for the worse performance of the new sources compared to the original source; e.g., in German there are rarely used subjunctive forms like

encoders: par’s shared:	$k = 1$		$k = 4$	
			yes	no
ar	.944	.944	.920	
de	.980	.980	.975	
tu	.985	.983	.969	

Table 4: Accuracy of different architectures for the dataset with 4 source forms being available for prediction. The best result for each row is in bold.

befähle that are unlikely to help generate related forms that are more frequent.

The main result of the experiment is that multi-source MRI performs better than single-source MRI for all languages except for Hungarian and that, clearly, the more sources the better: using four sources is always better than using two sources. This result confirms our hypothesis, illustrated in Figure 1, that for most languages, different source forms provide complementary information when generating a target form and thus performance of the multi-source model is better than of the single-source model. Table 3 demonstrates that the two configurations we identified as promising for multi-source MRI, SINGLEFORM and MULTIFORM, occur frequently enough to boost the performance for seven of the eight languages, with the largest gains observed for Arabic (7.3%) and Russian (3.5%) and the smallest for Spanish (0.9%) and Georgian (1.3%) (comparing using source form 1 with using source forms 1–4).

Hungarian is the only language for which performance decreases, by a small amount (0.3%). We attribute this to overfitting: the multi-source model has a larger number of parameters, so it is more prone to overfitting. We would expect the performance to be the same in a comparison of two models that have the same size.

Error Analysis. We compare errors of single-source and multi-source models for German on development data. Most mistakes of the multi-source model are stem-related: *versterbst* for *verstirbst*, *erwerben* for *erwürben*, *Apfelsinenbaume* for *Apfelsinenbäume*, *lungenkränkes* for *lungenkrankes* and *übernehmte* for *übernahme*. In most of these cases, the stem of the lemma was used, which is correct for some forms, but not for the form that had to be generated. In one case, the multi-source model did not use the correct inflection rule: *braucht* for *gebraucht*—the inflectional rule that the past participle is formed by *ge-* was not applied.

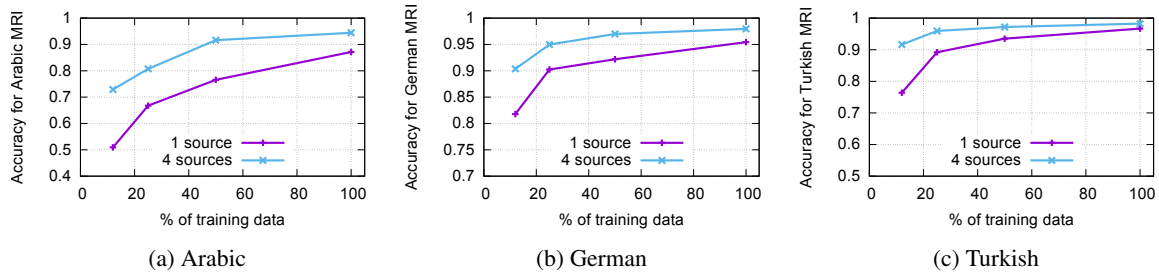


Figure 3: Learning curves for single-source and multi-source models for Arabic, German and Turkish. We observe that the multi-source model generalizes faster than the single source case—this is to be expected since the multi-source model often faces an easier transduction problem.

Errors of the single-source model that were “corrected” by the multi-source model include *emp-fahlt* for *empfehl*, *Throne* for *Thron* and *befielen* for *befallen*. These are all SINGLEFORM cases: the multi-source model will generate the correct form if it succeeds in selecting the most predictive source form. The single-source model is at a disadvantage if this most predictive source form is not part of its input.

4.3 Comparison of Different Architectures

Table 4 compares different architectural configurations. All experiments use 4 sources. We see that sharing parameters is superior as expected. Using a single encoder on 4 sources performs as well as 4 encoders (and very slightly better on Turkish). Apparently, it has no difficulty learning to understand an unstructured (or rather lightly structured) concatenation of form-tag pairs; on the other hand, this parsing task, i.e., learning to parse the sequence of form-tag pairs, is easy, so this is not a surprising result.

4.4 Learning Curves

Figure 3 shows learning curves for Arabic, German and Turkish. We iteratively halve the training set and train models for each subset. In this analysis, we train all models for 90 epochs, but use the numbers from the main experiment for the full training set. For the single-source model, we use the SIGMORPHON source. The figure shows that the single-source model needs more individual paradigms in the training data to achieve the same performance as the multi-source model. The largest difference between single-source and multi-source is $> 20\%$ for Arabic when only 1/8 of the training set is used. This suggests that multi-source MRI is an attractive option for low-resource languages since it exploits available data better than single-source.

4.5 Attention Visualization

Figure 4 shows for one example, the generation of the German form *wögen*, 3rdPISubPst, the attention weights of the multi-source model at each time step of the decoder, i.e., for each character as it is being produced by the decoder. For characters that simply need to be copied, the main attention lies on the corresponding characters of the input sources. For example, the character *g* is produced when attention is on the characters *g* in *wögest*, *wöge* and *wogen*. This aspect of the multi-source model is not different from the single-source model, offering no advantage.

However, even for *g*, the source form that is least relevant for generating *wögen* receives almost no weight: *wägst* is an indicative singular form that does not provide helpful information for generating a plural form in the subjunctive; the model seems to have learned that this is the case. In contrast, *wogen* does receive some weight; this makes sense as it is a past indicative form and the past subjunctive is systematically related to the past indicative for many German verbs. These observations suggest that the network has learned to correctly predict (at least in this case) which forms provide potentially useful information. For the last two time steps (i.e., characters to be generated), attention is mainly focused on the tags. Again, this indicates that the model has learned the regularity in generating this part of the word form: the suffix, consisting of *en*, is predictable from the tag.

5 Related Work

Recently, variants of the RNN encoder-decoder have seen widespread adoption in many areas of NLP due to their strong performance. Encoder-decoders with and without attention have been applied to tasks such as machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et

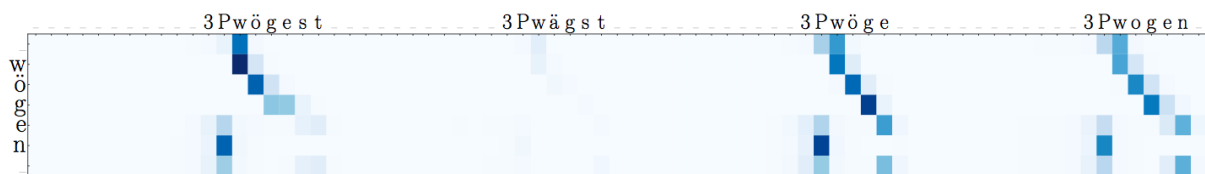


Figure 4: Attention heatmap for the multi-source model. The example is for the German verb *wiegen* ‘to weigh’. The model learns to focus most of its attention on forms that share the irregular subjunctive stem *wög* in addition to the target subtags *3* and *P* that encode that the target form is 3rd person plural. We omit the tags from the diagram to which the model hardly attends.

al., 2015), parsing (Vinyals et al., 2014) and automatic speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013).

The first work on multi-source models was presented for machine translation. Zoph and Knight (2016) made simultaneous use of source sentences in multiple languages in order to find the best match possible in the target language. Unlike our model, they apply transformations to the hidden states of the encoders that are input to the decoder. Firat et al. (2016)’s neural architecture for MT translates from any of N source languages to any of M target languages, using language specific encoders and decoders, but sharing one single attention-mechanism. In contrast to our work, they obtain a single output for each input.

Much ink has been spilled on morphological reinflection over recent years. Dreyer et al. (2008) develop a high-performing weighted finite-state transducer for the task, which was later hybridized with an LSTM (Rastogi et al., 2016). Durrett and DeNero (2013) apply a semi-CRF to heuristically extracted rules to generate inflected forms from lemmata using data scraped from Wiktionary. Improved systems for the Wiktionary data were subsequently developed by Hulden et al. (2014), who used a semi-supervised approach, and Faruqui et al. (2016), who used a character-level LSTM. All of the above work has focused on the single input case. Two important exceptions, however, have considered the multi-input case. Both Dreyer and Eisner (2009) and Cotterell et al. (2015b) define a string-valued graphical model over the paradigm and apply the missing values.

The SIGMORPHON 2016 Shared Task on Morphological Reinflection (Cotterell et al., 2016a), based on the UNIMORPH (Sylak-Glassman et al., 2015) data, resulted in the development of numerous methods. RNN encoder-decoder models (Aharoni et al., 2016; Kann and Schütze, 2016a; Östling, 2016) obtained the strongest performance and are the current state of the art on the task. The best-

performing model made use of an attention mechanism (Kann and Schütze, 2016a), first popularized in machine translation (Bahdanau et al., 2015). We generalize this architecture to the multi-source case in this paper for the reinflection task.

Besides generation, computational work on morphology has also focused on analysis. In this area, a common task—morphological segmentation—is to break up a word into its sequence of constituent morphs. The unsupervised MORFESSOR model (Creutz and Lagus, 2002) has achieved widespread adoption. Bayesian methods have also proven themselves successful in unsupervised morphological segmentation (Johnson et al., 2006; Goldwater et al., 2009). When labeled training data for segmentation is available, supervised methods significantly outperform the unsupervised techniques (Ruokolainen et al., 2013; Cotterell et al., 2015a; Cotterell et al., 2016b).

As we pointed out in Section 2, morphologically annotated corpora provide an ideal source of data for the multi-source MRI task: they are annotated on the token level with inflectional features and often contain several different inflected forms of a lemma. Eskander et al. (2013) develop an algorithm for automatic learning of inflectional classes and associated lemmas from morphologically annotated corpora, an approach that could be usefully combined with our multi-source MRI framework.

6 Conclusion

Generation of unknown inflections in morphologically rich languages is an important task that remains unsolved. We provide a new angle on the problem by considering systems that are allowed to have multiple inflected forms as input. To this end, we define the task of multi-source morphological reinflection as a generalization of single-source MRI (Cotterell et al., 2016a) and present a model that solves the task. We extend an attention-based RNN encoder-decoder architecture from the single-source case to the multi-source case. Our

new model consists of multiple encoders, each receiving one of the inputs. Our model improves over the state of the art for seven out of eight languages, demonstrating the promise of multi-source MRI. Additionally, we publically release our implementation.⁸

7 Future Work

The new dataset for multi-source morphological reinflection that we release is a superset of the dataset of the SIGMORPHON 2016 Shared Task on Morphological Reinflection to facilitate research on morphological generation. One focus of future work should be the construction of more complex datasets, e.g., datasets that have better coverage of irregular words and datasets in which there is no overlap in lemmata between training and test sets. Further, for difficult inflections, it might be interesting to find an effective way to include unsupervised data into the setup. For example, we could define one of our k inputs to be a form mined from a corpus that is not guaranteed to have been correctly tagged morphologically, but likely to be helpful.

We show in this paper that multi-source MRI outperforms single-source MRI. This is an important contribution because—as we discussed in Section 2.1—multi-source MRI is only promising for paradigms with specific properties, which we referred to as SINGLEFORM and MULTIFORM configurations. Whether such configurations occur and whether these configurations have a strong effect on MRI performance was an open empirical question. Indeed, we found that for one of the languages we investigated, for Hungarian, single-source MRI works at least as well as multi-source MRI—presumably because its paradigms almost exclusively contain SINGLEFORM configurations. Thus, single-source MRI is probably preferable for Hungarian since single-source is simpler than multi-source.

There is another important question that we have not answered in this paper: in an experimental setting in which the amount of training information available is exactly the same for single-source and multi-source, does multi-source still outperform single-source and by how much? For example, the numbers we compare in Table 3 are matched with respect to the number of target forms, but not with respect to the number of source forms: multi-

source has more source forms available for training than single-source. We leave investigation of this important issue for future work.

Acknowledgments

We gratefully acknowledge the financial support of Siemens and of DFG (SCHUE 2246/10-1) for this research. The second author was supported by a DAAD Long-Term Research Grant and an NDSEG fellowship.

References

- Roei Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 41–48, Berlin, Germany, August. Association for Computational Linguistics.
- Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1024–1029, Denver, Colorado, May–June. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, San Diego, California, USA, May.
- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.
- Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August. Association for Computational Linguistics.
- Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties

⁸<http://cistern.cis.lmu.de>

- of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October. Association for Computational Linguistics.
- Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.
- Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015a. Labeled morphological segmentation with semi-markov models. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 164–174, Beijing, China, July. Association for Computational Linguistics.
- Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015b. Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, 3:433–447.
- Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany, August. Association for Computational Linguistics.
- Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016b. A joint model of orthography and morphological segmentation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 664–669, San Diego, California, June. Association for Computational Linguistics.
- Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics, July.
- Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 101–110, Singapore, August. Association for Computational Linguistics.
- Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii, October. Association for Computational Linguistics.
- Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.
- Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia, June. Association for Computational Linguistics.
- Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic extraction of morphological lexicons from morphologically annotated corpora. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1032–1043, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Manaal Faruqi, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 634–643, San Diego, California, June. Association for Computational Linguistics.
- Raphael Finkel and Gregory Stump. 2007. Principal parts and morphological typology. *Morphology*, 17(1):39–75.
- Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, June. Association for Computational Linguistics.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.
- Alex Graves and Jürgen Schmidhuber. 2005. Frame-wise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.
- Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, Vancouver, BC, Canada, May.
- Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578, Gothenburg, Sweden, April. Association for Computational Linguistics.
- Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for

- specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems 19*, pages 641–648, Vancouver, BC, Canada, December.
- Katharina Kann and Hinrich Schütze. 2016a. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological inflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 62–70, Berlin, Germany, August. Association for Computational Linguistics.
- Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for inflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany, August. Association for Computational Linguistics.
- Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.
- Ryan McDonald, Joakim Nivre, Yvonne Quirnbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Thomas Müller, Ryan Cotterell, Alexander M. Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal, September. Association for Computational Linguistics.
- Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 922–931, Denver, Colorado, May–June. Association for Computational Linguistics.
- Robert Östling. 2016. Morphological inflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 23–26, Berlin, Germany, August. Association for Computational Linguistics.
- Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California, June. Association for Computational Linguistics.
- Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 29–37, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Gregory Stump and Raphael A. Finkel. 2013. *Morphological typology: From word to paradigm*, volume 138. Cambridge University Press.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112, Montreal, Quebec, Canada, December.
- John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 674–680, Beijing, China, July. Association for Computational Linguistics.
- Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2014. Grammar as a foreign language. *CoRR*, abs/1412.7449.
- Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June. Association for Computational Linguistics.

Online Automatic Post-editing for MT in a Multi-Domain Translation Environment

Rajen Chatterjee^(1,2), Gebremedhen Gebremelak^(1,2), Matteo Negri⁽²⁾, Marco Turchi⁽²⁾

⁽¹⁾University of Trento

⁽²⁾Fondazione Bruno Kessler

{chatterjee, gebremelak, negri, turchi}@fbk.eu

Abstract

Automatic post-editing (APE) for machine translation (MT) aims to fix recurrent errors made by the MT decoder by learning from correction examples. In controlled evaluation scenarios, the representativeness of the training set with respect to the test data is a key factor to achieve good performance. Real-life scenarios, however, do not guarantee such favorable learning conditions. Ideally, to be integrated in a real professional translation workflow (e.g. to play a role in computer-assisted translation framework), APE tools should be flexible enough to cope with continuous streams of diverse data coming from different domains/genres. To cope with this problem, we propose an online APE framework that is: *i*) robust to data diversity (i.e. capable to learn and apply correction rules in the right contexts) and *ii*) able to evolve over time (by continuously extending and refining its knowledge). In a comparative evaluation, with English-German test data coming in random order from two different domains, we show the effectiveness of our approach, which outperforms a strong batch system and the state of the art in online APE.

1 Introduction

Automatic post-editing (APE) systems for machine translation (MT) aim to correct the errors present in a machine-translated text before showing it to the user, thereby reducing human workload and eventually increase translation productivity. The choice of having such post-processing systems is well motivated in (Bojar et al., 2015) and becomes a must when the MT engine used

to translate is not directly accessible for retraining or for more radical internal modifications (e.g. when working with a third party MT system). As pointed out by (Parton et al., 2012; Chatterjee et al., 2015b), from the application point of view an APE system can help to: *i*) Improve MT output by exploiting information unavailable to the decoder, or by performing deeper text analysis that is too expensive at the decoding stage; *ii*) Provide professional translators with improved MT output quality to reduce (human) post-editing effort and *iii*) Adapt the output of a general-purpose MT system to the lexicon/style requested in a specific application domain. Similar to what is usually done in MT, APE components learn post-editing rules from “parallel” corpora consisting of machine-translated text (*mt*, optionally with its corresponding source text – *src*) and its post-edits (*pe*) provided by human post-editors. The effectiveness of learning from relatively small amounts of post-edited data is evident from the impressive outcomes of the recently held APE shared task at WMT 2016 (Bojar et al., 2016). Different APE paradigms, like neural (Junczys-Dowmunt and Grundkiewicz, 2016), hybrid (Chatterjee et al., 2016), and phrase-based (Pal et al., 2016) were all able to significantly improve MT output quality in the IT domain, with gains ranging from 2.0 to 5.5 BLEU points. Nevertheless, this success and the positive outcomes of previous work on automatic MT error correction build on a problem formulation that assumes to operate in a controlled lab environment, where the systems are trained and evaluated across a coherent/homogeneous data set. Moving from this controlled scenario to real-world translation workflows, where training and test data can be produced by different MT systems, post-edited by various translators and belong to several text genres, makes the task inherently more challenging, because the APE systems have to adapt to

all these diversities in real-time. In addition to the problem that training data provide a fraction of the possible error correction examples (a normal issue when learning from finite, often small training data), the additional complexity derives from two concurrent factors. First, not all the learned error correction rules are universally applicable: applying them in the wrong context can damage the MT output instead of improving it. Second, once in production, the APE system should be able to process streams of diverse input data presented in random order: promptly reacting to such variability is hence crucial. We define this more complex and realistic scenario as a *multi-domain* translation environment (MDTE), where a domain is made of segments belonging to the same text genre and the MT outputs are generated by the same MT system.

To our knowledge, the evaluation of APE systems on MDTE data in real-time/online translation scenarios is still unexplored. This paper represents a first step along this direction: although a full-fledged evaluation centered on human translation in a computer-assisted translation (CAT) framework is out of our reach, we provide a proof of concept in which we simulate the MDTE scenario by running different APE solutions on a stream of data coming from two different domains. By analysing alternative solutions, we discuss the limitations not only of batch APE methods (insensitive to domain shifts), but also of state-of-the-art online translation systems evaluated in the APE task in MDTE conditions. Thot (Ortiz-Martinez and Casacuberta, 2014), the online system used as term of comparison, shows in fact the inability to discern which of the learned correction rules is suitable for a specific context. In practice, all rules are created equal, for any given domain.

To overcome this limitation, we proceed incrementally. First, we propose an approach based on an instance selection strategy, which learns local, sentence-specific APE models from small amounts of relevant data for each translation to be post-edited. Then, on top of it, we add an improved way to estimate the parameters of the sentence-specific APE models. To this aim, we exploit a dynamic knowledge base that keeps track of global statistics computed over all the previously seen data (i.e. it does not rely only on those computed from the selected instances). Finally, the dynamic knowledge base gives us the possibility to experiment with new features in addition to those

used by current APE systems based on the phrase-based MT paradigm. Such features incorporate in the translation models also the negative feedback collected from human post-editors. Instead of continuously expanding our knowledge base of correction rules (i.e. only considering the positive feedback about how to correct a given error), we also stepwise refine it by weighing the acquired correction rules according to their reliability (e.g. demoting those that led to corrections eventually modified by the human). Positive evaluation results reflect this incremental approach. To summarize, our contribution is a fully automated online APE system that does not rely on pre-trained models or tuned weights (unlike Thot that needs to be pre-trained and tuned) and incorporates for the first time both positive and negative post-editors' feedback to set the state-of-the-art in the difficult task of APE in the MDTE scenario.

2 Related work

Most of the previous works on APE cast the problem as a phrase-based statistical MT task¹ and operate in a batch framework where systems are evaluated on static test sets that are homogeneous with the training data (Simard et al., 2007; Dugast et al., 2007; Terumasa, 2007; Pilevar, 2011; Béchara et al., 2011; Chatterjee et al., 2016). These systems, however, are not able to leverage the feedback of the post-editors in an online translation scenario. The capability to evolve by learning from human feedback has been addressed by several online translation systems but mainly focusing on the MT task (Hardt and Elming, 2010; Bertoldi et al., 2013; Mathur et al., 2013; Simard and Foster, 2013; Ortiz-Martinez and Casacuberta, 2014; Denkowski et al., 2014; Wuebker et al., 2015). From these several online MT systems, we discuss the two that have been used also for the APE task.

PEPr: Post-Edit Propagation. (Simard and Foster, 2013) proposed a method for post-edit propagation (PEPr), which learns post-editors' corrections and applies them on-the-fly to an MT output sequence. To perform post-edit propagation, the system is trained incrementally us-

¹Only recently, the wave of neural models has also reached the APE task (Junczys-Dowmunt and Grundkiewicz, 2016), setting the new state of the art at WMT (Bojar et al., 2016). The problem addressed in this paper (dealing with MDTE data), as well as the proposed online solution are still too computationally intensive to experiment with neural models. We hence leave this aspect as a future work direction.

ing pairs of machine-translated (*mt*) and human post-edited (*pe*) segments as they were produced. When receiving a new (*mt*, *pe*) pair, word alignments are obtained using Damerau-Levenshtein distance. In the next step, the phrase pairs are extracted and appended to the existing phrase table. The whole process is assumed to take place within the context of a single document and, for every new document, the APE system is initialised with an “empty” model. This represents a possible limitation of the approach: although document-specific correction rules show a relatively high precision, some of them might in fact be useful also in other contexts and should be retained. Our approach avoids this limitation by maintaining a global knowledge base to store all the processed documents, still being able to retrieve post-editing rules specific to a document to be translated.²

Thot. The Thot toolkit (Ortiz-Martinez and Casacuberta, 2014) is developed to support automatic and interactive statistical machine translation.³ It was also successfully used by Lagarda et al. (2015) to experiment in an online APE setting with several data sets for multiple language pairs, with base MT systems built using different technologies (rule-based MT, statistical MT). In order to incorporate user feedback in the underlying translation and language models, the systems maintains and incrementally updates all the required statistics. For the language model, it simply updates n-gram counts. In the case of the translation model, the process exploits an incremental version of expectation maximization algorithm to obtain word alignments and extract the phrase pairs whose counts are continuously updated. Other features, like source/target phrase-length models or the distortion model, are extracted considering geometric distributions with fixed parameters. The feature weights of the log-linear model are static throughout the online learning process, as opposed to our method that updates the weights on-the-fly. This makes our online APE approach independent from any pre-trained model or pre-tuned feature weights. Moreover, while in Thot the correction rules are learned in real-time from all the data processed, our system only learns from the most relevant data samples. Neverthe-

less, considering Thot as the state-of-the-art in online APE, we will use it as a term of comparison in our experiments.

3 Online APE system

The backbone of our online APE system is similar to the state-of-the-art statistical batch APE approach proposed in (Chatterjee et al., 2015b). The system is trained on (*src*, *mt*, *pe*) triplets, and learns correction rules in the form of (*mt*#*src*, *pe*) pairs. The first element of each pair consists of MT phrases (single or multiple words) that are associated to their corresponding source words by using a word alignment model. This “joint representation” helps to restrict the applicability of each rule to the appropriate context, and was shown to perform better than using only the *mt* words as the left-hand side of the rules (Béchara et al., 2011). Our migration to the online scenario builds on incrementally extending this backbone with an instance selection mechanism (§3.1), a dynamic knowledge base (§3.2) and new features (§3.3).

3.1 Instance selection

Current batch and online APE methods estimate parameters of the models over all the available training data. This strategy may not be effective in the MDTE scenario, since the model will tend to become more and more generic by incorporating knowledge from several domains. In the long-run, this can complicate the selection of domain-specific correction rules suitable for a particular MT segment. One of the possible solutions is to constrain the system to work at document level as proposed by Simard and Foster (2013). In their approach, however, the models are reset back to their original state once the entire document is processed, due to which knowledge gained from the current document is lost. Our instance selection technique aims to overcome this issue, allowing the system to preserve all the knowledge acquired during the online learning process, still being able to apply specific post-editing rules when needed.

The instance selection mechanism consists in retrieving *ad-hoc* training sentence pairs for each MT output to be post-edited. In practice, the creation of the APE model and the estimation of its parameters are performed on-the-fly by processing relevant instances retrieved from the previously processed data. In the MDTE scenario, this will come from heterogeneous domains. The rele-

²In our experiments we do not compare against PEP since, being designed for document-level translation it is unable to operate in the MDTE scenario.

³<https://github.com/daormar/thot>

vance of a training sample is measured in terms of a similarity score based on term frequency-inverse document frequency (tf-idf⁴) computed using the Lucene software library.⁵ Indexing and retrieving training triplets (*src*, *mt*, and *pe*) in this mechanism is fast, which makes it perfectly suitable to use in an online learning scenario. The cut-off similarity score is empirically estimated over a held-out development set. Input segments not having training samples above the threshold are left untouched to avoid any possible damage resulting from the application of unreliable correction rules learned from unrelated contexts. This is in contrast with the strategy adopted by current APE systems, which tend to always “translate” the given input segment, independently from the reliability of the applicable correction rules.

Once the training samples are selected for an input segment, several models are built on-the-fly. A tri-gram local language model (LM) is built over the target side of the training corpus with the IRSTLM toolkit (Federico et al., 2008). Along with the local LM a tri-gram global LM is also used, which is updated whenever a human post-edition (*pe*) is received. Local translation and reordering models are built with the Moses toolkit (Koehn et al., 2007), computing word alignment for each sentence pair using the incremental GIZA++ software.⁶

The feature weights of the log-linear model are optimized over a subset of the selected instances. The size of this development set is critical: if it is too large, then parameter optimization will be expensive. On the other hand, if it is too small, the tuned weights might not be reliable. To achieve fast optimization with reliably-tuned weights, multiple instances of MIRA (Crammer and Singer, 2003) are run in parallel on multiple development sets (Tange, 2011). For this purpose, the retrieved samples are randomly split three times into training and development. The tuned weights resulting from the three development runs are then averaged and used to decode the input MT segment.

To summarize, our training/tuning scheme requires a minimum number of retrieved sentence

⁴In MT, tf-idf was previously used by Hildebrand et al. (2005) to create a pseudo in-domain corpus from a large out-of-domain corpus. Our work is the first to investigate it for the APE task in an online learning scenario.

⁵<https://lucene.apache.org/>

⁶<https://code.google.com/archive/p/inc-giza-pp/>

pairs. Following an 80%-20% distribution over training and development data, and setting to 5 the minimum number of samples needed for tuning, the complete process requires the retrieval of at least 25 samples. If this number is not reached, all the retrieved samples are used for training, the optimization step is skipped and the previously computed weights are used. If no sample is selected, then the MT output will be left untouched.

3.2 Dynamic knowledge base

The APE system described so far is built by considering only the most similar retrieved sentences, which we hypothesize to be the most useful to learn reliable corrections for a given MT output. On one hand, this strategy avoids to end up with correction options that are not appropriate to post-edit the MT output. On the other hand, it computes the statistics of the models (i.e. translation and lexicalized reordering probabilities) using only few parallel sentences, resulting in potentially unreliable values that can penalise the work of the decoder. To address this issue, we complement instance selection with a dynamic knowledge base able to keep track of all the previous observations relevant for post-editing. Such a component provides the decoder with all the translation options extracted from the retrieved sentences but, instead of computing the probabilities only on these segments, it takes advantage of all the occurrences of a translation option in the previously processed sentences. This allows our online APE system to use only the most useful translation options, associated with more reliable statistics.

The dynamic knowledge base is implemented by a distributed, scalable and real-time inverted index that, after insertion, makes all data immediately available for search and update. The ElasticSearch⁷ engine is used for this purpose. Once the post-edit is made available to our system, the word alignment between the *mt* and the *pe* is computed, the sentence pair is split in phrases and then added to the dynamic model. If a translation option is already present, then the phrase translation and the orientation counts are updated, otherwise it is inserted for the first time. This is run in multi-threading, by also managing possible conflicts (i.e. the access to the same translation option by different threads). Word lexical information and phrase

⁷<http://www.elastic.co/products/elasticsearch>

counts are stored apart. At decoding time, the IDs of the samples retrieved by instance selection and the *mt* sentences are used to query the dynamic knowledge base. The translation options that satisfy the query are retrieved and supplied to the decoder in the form of translation and reordering model information. All the feature scores (four for the translation model and six for the reordering model) are computed on-the-fly.

Compared to the suffix arrays used to implement MT dynamic models (Germann, 2014; Denkowski et al., 2014), in which the whole sentence pairs are stored, our technique needs to save more information (all the translation options) but: *i*) the amount of data in APE is much less than in MT so it can be easily managed by *ad hoc* solutions, and *ii*) it allows us to collect global information at translation option level that can result in useful additional features for the model. This last aspect is explored in the next section, in which the reliability of the translation options is measured by looking at the behavior of the post-editors.

3.3 Negative feedback in-the-loop

Similar to the APE systems mentioned in Section 2, the one described so far stores only post-editors' positive feedback. Its knowledge base of correction rules and the statistics to estimate the model parameters are in fact continuously updated only based on alignment information between (*mt*, *pe*) pairs. Post-edits, however, can also be used to infer negative feedback and use it to penalize unreliable correction options (i.e. those resulting in post-edits eventually modified by the human). The dynamic knowledge base allows us to easily integrate this kind of information, in the form of two additional "negative feedback" features:

- F1. This feature penalizes the correction rules that are selected by the decoder but eventually modified by the post-editor. This can be due to the application of a rule in the wrong context (e.g. in case of domain changes in the input stream of data) but, most likely, to the fact that the learned rule is wrong (e.g. as the result of ambiguous/wrong word alignment). It is computed as the ratio of the number of times the post-editors modified a correction made by the APE decoder to the total number of times the decoder has made the correction. The information about which correction rules have been

applied by the APE system is obtained from the Moses decoder `trace` option. The information about which of them has been modified is derived by string matching the target side of the rule in the final human post-edit.

- F2. This rule penalizes the correction rules that, even if not used, were available to the decoder (i.e. translation options discarded during decoding). Assuming that the application of these options would have been eventually corrected by the post-editor, all the rules in the phrase table are scanned to check if their target side (i.e. the correction) is present in the final human post-edit (again by string matching). If not, then the corresponding rule is penalised. This feature is computed as the ratio of the number of times the correction in the phrase table is (assumed to be) modified by the post-editor to the total number of time the correction rule has been seen in the local phrase table for all the segments processed so far. Since the decoder operates with a segment-specific local phrase table containing only the options relevant to the segment to be post-edited, computing this feature is not expensive.

We also evaluate system performance by using the two features together. As we will see in Section 5, although our use of negative feedback is still at a preliminary stage, its integration in our online APE framework yields some improvements.

4 Evaluation setting

Data. We experiment with two English-German data sets: *i*) the data released for the APE shared task organised within the first Conference on Machine Translation (WMT16) (Bojar et al., 2016), and *ii*) the data used in (Chatterjee et al., 2015b), which is a subset of the Autodesk Post-Editing Data corpus.⁸ Although they come from the same category (IT), they feature variability in terms of vocabulary, MT engines used for translation, MT errors and post-editing style. According to our broad notion of "domain", these variations contribute to make the two data sets different enough to emulate an MDTE scenario for testing online APE capabilities. The data are pre-processed to obtain a joint representation that

⁸<https://autodesk.app.box.com/v/autodesk-postediting>

	Tokens		Types		Avg. segment length		RR (<i>mt#src</i>)	TER
	<i>mt#src</i>	<i>pe</i>	<i>mt#src</i>	<i>pe</i>	<i>mt#src</i>	<i>pe</i>		
Autodesk	153,943	160,801	31,939	15,023	12.57	13.13	4.938	45.35
WMT16	210,573	214,720	32211	16,388	17.54	17.89	4.907	26.22

Table 1: Data statistics

links each MT word with its corresponding source word/s (*mt#src*). This representation, proposed by Béchara et al. (2011), leverages the source information to disambiguate post-editing rules and foster their application only in appropriate contexts (the matching condition is defined both on the source and on the target language). The joint representation is used as a source corpus to train all the APE systems compared in this paper and it is obtained by concatenating words in the source (*src*) and in the MT (*mt*) segments after aligning them with MGIZA++ (Gao and Vogel, 2008).

The Autodesk training and development sets consist of 12,238, and 1,948 segments respectively, while the WMT16 data contains 12,000, and 1,000 segments. Table 1 provides additional statistics of the source (*mt#src*) and target (*pe*) training sets, the repetition rate (RR) to measure the repetitiveness inside a text (Bertoldi et al., 2013), and the average TER score for both the data sets (computed between MT and PE), as an indicator of the original translation quality. Looking at these statistics, there are several indicators that suggest that the WMT16 corpus provides a more difficult scenario for APE than the Autodesk one. First, the WMT16 corpus has on average longer sentences, which generally increases the complexity of the rule extraction and decoding processes. Second, although the two data sets have a similar repetition rate, the WMT16 has more tokens indicating the higher sparsity of the data. Finally, the lower TER of WMT16 suggests that there are less corrections to perform and, in turn, a higher chance to deteriorate the original MT output if the learned rules are applied in the wrong context.

To conclude, we measure the diversity of the two data sets by computing the vocabulary overlap between the two joint representations. This is performed internally to each data set (splitting the training data in two halves) and across them. As expected, in the first case the vocabulary overlap is much larger ($> 40\%$) than in the second case ($\sim 15\%$) indicating that the information to share between the two data sets is minimal.

In our MDTE experiments, the training data is first merged, then shuffled and then split in two halves of 12,119 segments. The same procedure is applied to the development sets.

Evaluation metrics. The performance of the different systems is evaluated in terms of Translation Error rate (TER) (Snover et al., 2006), BLEU (Papineni et al., 2002), and precision (Chatterjee et al., 2015a). TER and BLEU measure the similarity between the MT output and the corresponding references (in this case human post-edits) by looking at the word/n-gram overlaps. Precision is the ratio of the number of sentences an APE system improves (with respect to the MT output) over all the sentences it modifies.⁹ Higher precision indicates that the APE system is able to improve the quality of most of the sentences it changed. The statistical significance of BLEU results is computed using paired bootstrap resampling (Koehn, 2004). For TER, we use stratified approximate randomization (Clark et al., 2011).

Terms of comparison. We evaluate our online learning approach against: *i*) the MT baseline (i.e. the MT output left untouched), *ii*) the batch APE system described in Section 3, on top of which we incrementally add our online learning extensions, and *iii*) the Thot toolkit.

5 Experiments and results

The batch APE system is trained on the first half of the shuffled training set, tuned with the development set (2,948 segments), and evaluated over the second half of the training data. Since the batch APE only learns from the training set, we expect its performance to give us a lower bound estimate, which should be outperformed by the online APE systems that can learn from the test data too. To run the online experiments with Thot, the system first needs to estimate the feature weights of the log-linear model on a develop-

⁹For each sentence in the test set, if the TER score of the APE system is different from the baseline then it is considered as a modified sentence.

ment set. For this purpose, it is trained and tuned off-line like a batch APE system. Three online extensions of the batch backbone architecture described in Section 3 are evaluated. These are: *i*) the instance selection system (IS); *ii*) the dynamic knowledge base system (IS+DynKB) and *iii*) the dynamic knowledge base system enhanced with the negative feedback features, both alone and in combination (IS+DynKB+F1, IS+DynKB+F2 and IS+DynKB+F1+F2). For all of them, the cut-off similarity score is obtained by grid search and is set to 0.8. The results achieved by each system are reported in Table 2.

	BLEU \uparrow	TER \downarrow	Prec. (%)
MT	52.31	34.52	N/A
Batch APE	52.52	34.45	42.67
Thot	52.51	34.37	42.22
IS	53.35 \dagger	33.36 \dagger	58.47
IS+DynKB	53.60 \dagger	33.23 \dagger	59.69
IS+DynKB+F1	53.56 \dagger	33.29 \dagger	58.97
IS+DynKB+F2	53.21 \dagger	33.48 \dagger	54.64
IS+DynKB+F1+F2	53.77\dagger	33.20\dagger	60.93

Table 2: Results on the mixed data. (“ \dagger ” indicates statistically significant difference wrt. the MT baseline with $p < 0.05$).

As can be seen from the table, the batch APE system is able to slightly improve over the baseline even if it damages most of the translations it changes (its precision is in fact lower than 45%). Although it learns also from the test data, Thot achieves similar results. This is probably due to its inability to identify domain-specific correction rules needed to improve the translations, thus ending up with damaging the majority of the modified MT segments. A significant gain in performance (+1.04 BLEU, -1.16 TER points) is obtained by our online IS system that, by using the instance selection technique, is able to isolate only the most useful training samples. This mechanism also improves precision up to 58.4% ($\sim 16\%$ above Thot), indicating that the applied post-editing rules are correct in the majority of the cases. The analysis of the performance of the two online systems reveals that our approach modifies less segments compared to Thot, due to the fact that it builds sentence-specific models only if it finds relevant data, leaving the MT segment untouched otherwise. In several cases, when modified by the Thot system, these untouched segments result in deterio-

rated sentences.

Further performance improvements are yielded by the dynamic knowledge base (IS+DynKB), which provides the decoder with a better estimation of the APE model parameters. Although the BLEU and TER gains are minimal, the dynamic knowledge base helps to significantly increase the precision of the APE system avoiding unnecessary changes, thus confirming the effectiveness of keeping track of the whole past history of each translation option. Our implementation of the dynamic knowledge base also allows us to add the two “negative feedback” features that model the reliability of the translation options by looking at the changes made by the post-editors. When used in combination, the two negative feedback features (IS+DynKB+F1+F2) yield visible gains in performance over (IS+DynKB) with small but statistically significant improvement in BLEU score, along with a precision gain of 1.24%. This suggests their possible complementarity with the translation and reordering features and the need of further investigation in future work. Overall our full-fledged system achieves state-of-the-art results with significant improvement over Thot by 1.26 BLEU, 1.17 TER, and 18.71% Precision.

	BLEU \uparrow	TER \downarrow	Prec. (%)
Autodesk			
MT	40.01	45.42	N/A
Batch APE	43.13 \dagger	43.19 \dagger	58.86
Thot	43.40 \dagger	42.96 \dagger	59.04
IS+DynKB+F1+F2	44.56\dagger	41.86\dagger	73.37
WMT16			
MT	61.04	26.24	N/A
Batch APE	59.24 \dagger	27.81 \dagger	22.18
Thot	59.05 \dagger	27.84 \dagger	20.06
IS+DynKB+F1+F2	60.39\dagger	26.62\dagger	36.67

Table 3: Performance analysis of each domain.

6 Analysis

To understand and compare the behavior of the batch APE, Thot and our best online system in the long-run, the plot in Figure 1 shows the TER moving average (window of 750 data points) at each segment of the test set (second half of the shuffled training data). As can be seen, our approach successfully maintains the best performance across the entire test set. Moreover, looking at the first

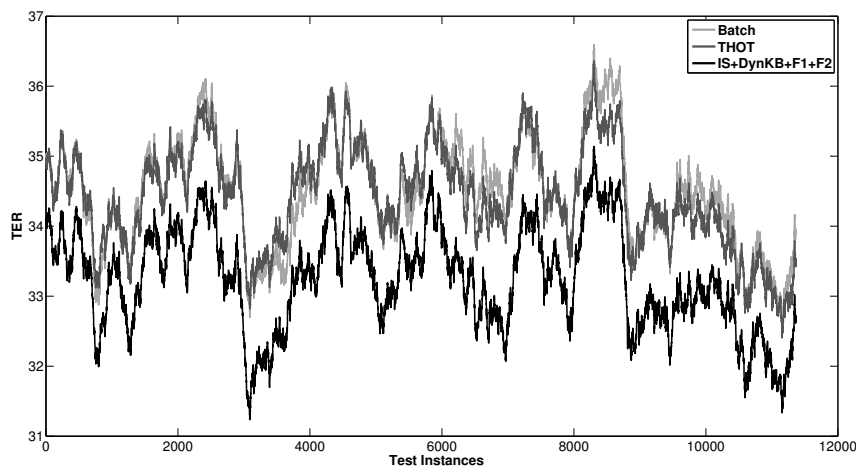


Figure 1: Performance comparison of different online APE systems (TER moving average).

SRC	Specifies the value to define the mid-ordinate distance by which to tessellate baseline alignment curves .
MT	Gibt den Wert für den krzesten Abstand vom Sekantenmittelpunkt zu Kreisbogen für die Tessellation Basislinienachse Kurven .
MT-Top1	Gibt den Wert für den krzesten Abstand vom Sekantenmittelpunkt zu Kreisbogen für die Tessellation Basislinienachse Kurven .
PE-Top1	Gibt den Wert zum Definieren des krzesten Abstands vom Sekantenmittelpunkt zum Kreisbogen an , um den Basislinienachsen-Bogen ausgerundet werden sollen .
THOT	Gibt den Wert für den Versatzzielbogen Abstand vom Sekantenmittelpunkt zu Kreisbogen für die Tessellation Basislinienachse Kurven .
IS+DynKB+F1+F2	Gibt den Wert zum Definieren des krzesten Abstands vom Sekantenmittelpunkt zum Kreisbogen an , um den Basislinienachse Versatzzielbogen ausgerundet werden sollen .
REF	Gibt den Wert zum Definieren des krzesten Abstands vom Sekantenmittelpunkt zum Kreisbogen an , um den Basislinienachsen-Bogen ausgerundet werden sollen .

Table 4: Sample outputs where our approach outperform THOT (erroneous words are in bold)

and the last 250 points in the test set, we notice that the performance gap between our best system and Thot increases on average from 0.8 to 1.6 TER points. This shows that, during processing, our approach is able to self-adapt in real-time towards the domain-shifts in the input stream of data. To better understand their behavior with respect to data coming from the two domains, systems' output has been separately evaluated per domain. The results of this evaluation are reported in Table 3.

For the Autodesk and the APE shared task domain there are 6,166 and 5,953 segments respectively. It is interesting to see that all the APE systems improve the translations belonging to the Autodesk domain by a large margin, with our approach being the best. The same does not hold for the other domain, which, as discussed in Sec-

tion 5, is more challenging due to several factors like longer sentence length, higher data sparsity and, most of all, lower translation quality. For this challenging domain, however, our approach has the least degradation compared to the other APE methods that severely damage the translations. Overall, compared to other APE approaches, our best system has the best performance in both the domains when evaluated in isolation.

To evaluate the efficiency of our approach, we computed the average time in seconds to perform a full online cycle over the test set (*i.e.* the time needed for post-editing the MT output and updating the models) for Thot, IS and IS+DynKB+F1+F2. Thot spends on average 4.75 seconds per cycle. The IS system, which builds its models on-the-fly by leveraging only the selected

SRC	Drag to the left and then click to place .
MT	Ziehen Sie nach links , und klicken Sie dann , um sie zu platzieren .
MT-Top1	Ziehen Sie nach links , und klicken Sie dann , um sie zu platzieren .
PE-Top1	Ziehen Sie sie nach links , und klicken Sie dann , um sie zu platzieren .
MT-Top4	Ziehen Sie den Cursor nach unten und nach rechts , und klicken Sie dann , um sie zu platzieren .
PE-Top4	Ziehen Sie nach unten und nach rechts , und klicken Sie dann zum Platzieren .
THOT	Ziehen Sie nach links , und klicken Sie dann , um sie zu platzieren .
IS+DynKB +F1+F2	Ziehen Sie nach links , und klicken Sie dann zum Platzieren .
REF	Ziehen Sie sie nach links , und klicken Sie dann , um sie zu platzieren .

Table 5: Sample output where our approach performs poor than THOT (erroneous words are in bold)

data and optimises the weights before post-editing, is faster than Thot, with a gain of 1.03 seconds (3.62" on average). The use of the dynamic model, that is faster in updating and dumping the tables, allows our best system to perform a full online cycle in 3.05", showing that our approach is not only better in terms of performance but also in computation time.

Tables 4 and 5 respectively show examples where our approach performs better/worse than Thot. Both tables report the source sentence (SRC), the MT output to be post-edited (MT), the MT and the PE segment of the top training samples retrieved based on cosine similarity (MT-TopX/PE-TopX, where X is the rank of the training sample), the output of Thot, the output of our best system (IS+DynKB+F1+F2) and the reference (REF). Table 4 seems to confirm our intuition that learning from the most similar examples yields better translation quality. An interesting counter example is shown in Table 5, where despite having access to a training sample (MT-Top1 and PE-Top1) that is exactly the same as the MT segment to be post-edited, our system deteriorates the translation by selecting a translation option ("zu platzieren" → "zum Platzieren") learned from a lower ranked training sample (MT-Top4 and PE-Top4), which probably received a higher weight from the local models. In future work, we will try to extend our system to address this issue by prioritizing the translation rules according to the rank of the training samples.

7 Conclusion

In recent years, APE systems achieved impressive results in fixing recurrent errors in machine-translated texts. Such gains, however, were mostly

observed in controlled lab environments, where systems are trained, tuned, and evaluated with repetitive and homogeneous training/test data. These favorable learning conditions may not hold in real-world professional translation workflow, in which streams of data to be processed in real-time may feature a high diversity in terms of domain, post-editing style and MT systems that generated the translations. In this paper, we investigated for the first time the challenges posed to APE technology by such multi-domain translation environments. Our study revealed that the existing online and batch solutions are not robust enough for this scenario due to their inability to discern which of the learned rules is suitable for a specific context (in fact, a correction rule learned from one domain may not be valid for other domains). We addressed this problem incrementally, first by proposing an instance selection technique that learns rules from contexts that are similar to the MT segment to be post-edited. The gains achieved by this solution over the existing batch APE methods were further increased by the addition of a dynamic knowledge base that stores more reliable statistics about the learned translation options, also improving the computation time. The adoption of this dynamic knowledge base allowed us to further extend our online approach by including features that capture negative human feedback, giving to the system the capability to learn from the mistakes it made in the past. Our evaluation results indicate that our approach improves state of the art performance on an English-German MDTE data set.

8 Acknowledgement

This work has been partially supported by the EC-funded H2020 project QT21 (grant no. 645452)

References

- Hanna Béchara, Yanjun Ma, and Josef van Genabith. 2011. Statistical Post-Editing for a Statistical MT System. In *Proceedings of the 13th Machine Translation Summit*, pages 308–315.
- Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2013. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. *Proceedings of the XIV Machine Translation Summit*, pages 35–42.
- Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September.
- Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August. Association for Computational Linguistics.
- Rajen Chatterjee, Marco Turchi, and Matteo Negri. 2015a. The FBK Participation in the WMT15 Automatic Post-editing Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 210–215.
- Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015b. Exploring the Planet of the APes: a Comparative Study of State-of-the-art Methods for MT Automatic Post-Editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 156–161, July.
- Rajen Chatterjee, José G. C. de Souza, Matteo Negri, and Marco Turchi. 2016. The FBK Participation in the WMT 2016 Automatic Post-editing Shared Task. In *Proceedings of the First Conference on Machine Translation*, pages 745–750, Berlin, Germany, August. Association for Computational Linguistics.
- Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 176–181.
- Koby Crammer and Yoram Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *The Journal of Machine Learning Research*, 3:951–991.
- Michael Denkowski, Chris Dyer, and Alon Lavie. 2014. Learning from Post-Editing: Online Model Adaptation for Statistical Machine Translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, April.
- Loïc Dugast, Jean Senellart, and Philipp Koehn. 2007. Statistical Post-Editing on SYSTRAN’s Rule-Based Translation System. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223.
- Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an Open Source Toolkit for Handling Large Scale Language Models. In *Proceedings of the Interspeech*, pages 1618–1621.
- Qin Gao and Stephan Vogel. 2008. Parallel Implementations of Word Alignment Tool. In *Proceedings of Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57.
- Ulrich Germann. 2014. Dynamic Phrase Tables for Machine Translation in an Interactive Post-editing Scenario. In *Proceedings of the Workshop on interactive and adaptive machine translation*, pages 20–31.
- Daniel Hardt and Jakob Elming. 2010. Incremental Re-training for Post-editing SMT. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.
- Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the Translation Model for Statistical Machine Translation based on Information Retrieval. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation*, pages 133–142.
- Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany, August. Association for Computational Linguistics.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, pages 177–180.
- Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the Empirical Methods on Natural Language Processing*, pages 388–395.

- Antonio L Lagarda, Daniel Ortiz-Martínez, Vicent Alabau, and Francisco Casacuberta. 2015. Translating without In-domain Corpus: Machine Translation Post-Editing with Online Learning Techniques. *Computer Speech & Language*, 32(1):109–134.
- Prashant Mathur, Mauro Cettolo, Marcello Federico, and FBK-Fondazione Bruno Kessler. 2013. Online Learning Approaches in Computer Assisted Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 301–308.
- Daniel Ortiz-Martinez and Francisco Casacuberta. 2014. The New THOT Toolkit for Fully-Automatic and Interactive Statistical Machine Translation. In *14th Annual Meeting of the European Association for Computational Linguistics*, pages 45–48.
- Santanu Pal, Marcos Zampieri, and Josef van Genabith. 2016. USAAR: An Operation Sequential Model for Automatic Statistical Post-Editing. In *Proceedings of the First Conference on Machine Translation*, pages 759–763, Berlin, Germany, August. Association for Computational Linguistics.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.
- Kristen Parton, Nizar Habash, Kathleen McKeown, Gonzalo Iglesias, and Adrià de Gispert. 2012. Can Automatic Post-Editing Make MT More Meaningful? In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation*, pages 111–118.
- Abdol Hamid Pilevar. 2011. Using Statistical Post-editing to Improve the Output of Rule-based Machine Translation System. *International Journal of Computer Science and Communication*.
- Michel Simard and George Foster. 2013. Pepr: Post-edit Propagation Using Phrase-based Statistical Machine Translation. In *Proceedings of the XIV Machine Translation Summit*, pages 191–198.
- Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical Phrase-Based Post-Editing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 508–515.
- Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 223–231.
- O. Tange. 2011. GNU Parallel - The Command-Line Power Tool. *login: The USENIX Magazine*, 36(1):42–47, Feb.
- Ehara Terumasa. 2007. Rule Based Machine Translation Combined with Statistical Post Editor for Japanese to English Patent Translation. In *Proceedings of the XI Machine Translation Summit*, pages 13–18.
- Joern Wuebker, Spence Green, and John DeNero. 2015. Hierarchical Incremental Adaptation for Statistical Machine Translation. In *Proceedings of the Empirical Methods in Natural Language Processing*, pages 1059–1065, September.

An Incremental Parser for Abstract Meaning Representation

Marco Damonte

School of Informatics
University of Edinburgh
m.damonte@sms.ed.ac.uk

Shay B. Cohen

School of Informatics
University of Edinburgh
scohen@inf.ed.ac.uk

Giorgio Satta

Dept. of Information Engineering
University of Padua
satta@dei.unipd.it

Abstract

Abstract Meaning Representation (AMR) is a semantic representation for natural language that embeds annotations related to traditional tasks such as named entity recognition, semantic role labeling, word sense disambiguation and co-reference resolution. We describe a transition-based parser for AMR that parses sentences left-to-right, in linear time. We further propose a test-suite that assesses specific sub-tasks that are helpful in comparing AMR parsers, and show that our parser is competitive with the state of the art on the LDC2015E86 dataset and that it outperforms state-of-the-art parsers for recovering named entities and handling polarity.

1 Introduction

Semantic parsing aims to solve the problem of canonicalizing language and representing its meaning: given an input sentence, it aims to extract a semantic representation of that sentence. Abstract meaning representation (Banarescu et al., 2013), or AMR for short, allows us to do that with the inclusion of most of the shallow-semantic natural language processing (NLP) tasks that are usually addressed separately, such as named entity recognition, semantic role labeling and co-reference resolution. AMR is partially motivated by the need to provide the NLP community with a single dataset that includes basic disambiguation information, instead of having to rely on different datasets for each disambiguation problem. The annotation process is straightforward, enabling the development of large datasets. Alternative semantic representations have been developed and stud-

ied, such as CCG (Steedman, 1996; Steedman, 2000) and UCCA (Abend and Rappoport, 2013).

Several parsers for AMR have been recently developed (Flanigan et al., 2014; Wang et al., 2015a; Peng et al., 2015; Pust et al., 2015; Goodman et al., 2016; Rao et al., 2015; Vanderwende et al., 2015; Artzi et al., 2015; Zhou et al., 2016). This line of research is new and current results suggest a large room for improvement. Greedy transition-based methods (Nivre, 2008) are one of the most popular choices for dependency parsing, because of their good balance between efficiency and accuracy. These methods seem promising also for AMR, due to the similarity between dependency trees and AMR structures, i.e., both representations use graphs with nodes that have lexical content and edges that represent linguistic relations.

A transition system is an abstract machine characterized by a set of configurations and transitions between them. The basic components of a configuration are a stack of partially processed words and a buffer of unseen input words. Starting from an initial configuration, the system applies transitions until a terminal configuration is reached. The sentence is scanned left to right, with linear time complexity for dependency parsing. This is made possible by the use of a greedy classifier that chooses the transition to be applied at each step.

In this paper we introduce a parser for AMR that is inspired by the ARCEAGER dependency transition system of Nivre (2004). The main difference between our system and ARCEAGER is that we need to account for the mapping from word tokens to AMR nodes, non-projectivity of AMR structures and reentrant nodes (multiple incoming edges). Our AMR parser brings closer dependency parsing and AMR parsing by showing that dependency parsing algorithms, with some mod-

ifications, can be used for AMR. Key properties such as working left-to-right, incrementality¹ and linear complexity further strengthen its relevance.

The AMR parser of Wang et al. (2015a), called CAMR, also defines a transition system. It differs from ours because we process the sentence left-to-right while they first acquire the entire dependency tree and then process it bottom-up. More recently Zhou et al. (2016) presented a non-greedy transition system for AMR parsing, based on ARC-STANDARD (Nivre, 2004). Our transition system is also related to an adaptation of ARCEAGER for directed acyclic graphs (DAGs), introduced by Sagae and Tsujii (2008). This is also the basis for Ribeyre et al. (2015), a transition system used to parse dependency graphs. Similarly, Du et al. (2014) also address dependency graph parsing by means of transition systems. Analogously to dependency trees, dependency graphs have the property that their nodes consist of the word tokens, which is not true for AMR. As such, these transition systems are more closely related to traditional transition systems for dependency parsing.

Our contributions in this paper are as follows:

- In §3 we develop a left-to-right, linear-time transition system for AMR parsing, inspired by the ARCEAGER transition system for dependency tree parsing;
- In §5 we claim that the Smatch score (Cai and Knight, 2013) is not sufficient to evaluate AMR parsers and propose a set of metrics to alleviate this problem and better compare alternative parsers;
- In §6 we show that our algorithm is competitive with publicly available state-of-the-art parsers on several metrics.

2 Background and Notation

AMR Structures AMRs are rooted and directed graphs with node and edge labels. An annotation example for the sentence *I beg you to excuse me* is shown in Figure 1, with the AMR graph reported in Figure 2.

Concepts are represented as labeled nodes in the graph and can be either English words (e.g. *I* and *you*) or Propbank framesets (e.g. *beg-01* and

¹Strictly speaking, transition-based parsing cannot achieve full incrementality, which requires to have a single connected component at all times (Nivre, 2004).

```
(b / beg-01
  :ARG0 (i / i
        :ARG1 (y / you
              :ARG2 (e / excuse-01
                    :ARG0 y
                    :ARG1 i)))
```

Figure 1: Annotation for the sentence “*I beg you to excuse me.*” Variables are in boldface and concepts and edge labels are in italics.

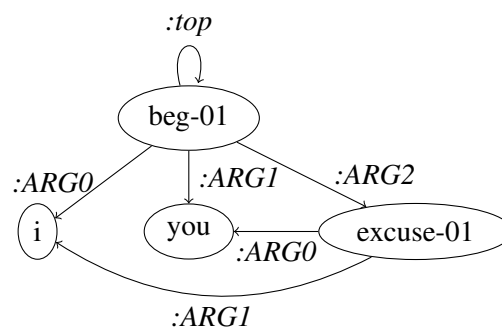


Figure 2: AMR graph representation for Figure 1.

excuse-01). Each node in the graph is assigned to a variable in the AMR annotation so that a variable re-used in the annotation corresponds to reentrancies (multiple incoming edges) in the graph. Relations are represented as labeled and directed edges in the graph.

Notation For most sentences in our dataset, the AMR graph is a directed acyclic graph (DAG), with a few specific cases where cycles are permitted. These cases are rare, and for the purpose of this paper, we consider AMR as DAGs.

We denote by $[n]$ the set $\{1, \dots, n\}$. We define an AMR structure as a tuple (G, x, π) , where $x = x_1 \dots x_n$ is a sentence, with each $x_i, i \in [n]$, a word token, and G is a directed graph $G = (V, E)$ with V and E the set of nodes and edges, respectively.² We assume G comes along with a node labeling function and an edge labeling function. Finally, $\pi: V \rightarrow [n]$ is a total alignment function that maps every node of the graph to an index i for the sentence x , with the meaning that node v represents (part of) the concept expressed by the word $x_{\pi(v)}$.³

We note that the function π is not invertible,

²We collapse all multi-word named entities in a single token (e.g., *United Kingdom* becomes *United_Kingdom*) both in training and parsing.

³ π is a function because we do not consider co-references,

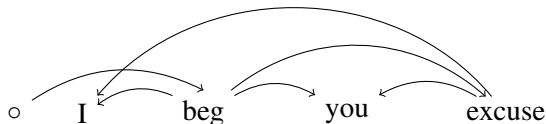


Figure 3: AMR’s edges for the sentence “I beg you to excuse me.” mapped back to the sentence, according to the alignment. \circ is a special token representing the root.

since it is neither injective nor surjective. For each $i \in [n]$, we let

$$\pi^{-1}(i) = \{v \mid v \in V, \pi(v) = i\}$$

be the pre-image of i under π (this set can be empty for some i), which means that we map a token in the sentence to a set of nodes in the AMR. In this way we can align each index i for x to the induced subgraph of G . More formally, we define

$$\overleftarrow{\pi}(i) = (\pi^{-1}(i), E \cap (\pi^{-1}(i) \times \pi^{-1}(i))), \quad (1)$$

with the node and edge labeling functions of $\overleftarrow{\pi}(i)$ inherited from G . Hence, $\overleftarrow{\pi}(i)$ returns the AMR subgraph aligned with a particular token in the sentence.

2.1 Transition-Based AMR Parsing

Similarly to dependency parsing, AMR parsing is partially based on the identification of predicate-argument structures. Much of the dependency parsing literature focuses on *transition-based* dependency parsing—an approach to parsing that scans the sentence from left to right in linear time and updates an intermediate structure that eventually ends up being a dependency tree.

Because of the similarity of AMR structures to dependency structures, transition systems are also helpful for AMR parsing. Starting from the ARCEAGER system, we develop here a novel transition system, called AMREAGER that parses sentences into AMR structures. There are three key differences between AMRs and dependency trees that require further adjustments for dependency parsers to be used with AMRs.

Non-Projectivity A key difference between English dependency trees and AMR structures is projectivity. Dependency trees in English are usually projective, roughly meaning that there are no

which would otherwise cause a node to map to multiple indices. This is in line with current work on AMR parsing.

Non-projective edges	6%
Non-projective AMRs	51%
Reentrant edges	41%
AMRs with at least one reentrancy	93%

Table 1: Statistics for non-projectivity and reentrancies in 200 AMR manually aligned with the associated sentences.⁵

crossing arcs if the edges are drawn in the semi-plane above the words. While this restriction is empirically motivated in syntactic theories for English, it is no longer motivated for AMR structures.

The notion of projectivity can be generalized to AMR graphs as follows. The intuition is that we can use the alignment π to map AMR edges back to the sentence x , and test whether there exist pairs of crossing edges. Figure 3 shows this mapping for the AMR of Figure 2, where the edge connecting *excuse* to *I* crosses another edge. More formally, consider an AMR edge $e = (u, \ell, v)$. Let $\pi(u) = i$ and $\pi(v) = j$, so that u is aligned with x_i and v is aligned with x_j . The spanning set for e , written $\mathcal{S}(e)$, is the set of all nodes w such that $\pi(w) = k$ and $i < k < j$ if $i < j$ or $j < k < i$ if $j < i$. We say that e is **projective** if, for every node $w \in \mathcal{S}(e)$, all of its parent and child nodes are in $\mathcal{S}(e) \cup \{u, v\}$; otherwise, we say that e is **non-projective**. An AMR is projective if all of its edges are projective, and is non-projective otherwise. This corresponds to the intuitive definition of projectivity for DAGs introduced in Sagae and Tsujii (2008) and is closely related to the definition of non-crossing graphs of Kuhlmann and Jonsson (2015).

Table 1 demonstrates that a relatively small percentage of all AMR edges are non-projective. Yet, a large fraction of the sentences contain at least one non-projective edge. Our parser is able to construct non-projective edges, as described in §3.

Reentrancy AMRs are graphs rather than trees because they can have nodes with multiple parents, called reentrant nodes, as in the node *you* for the AMR of Figure 2. There are two phenomena that cause reentrancies in AMR: control, where a reentrant edge appears between siblings of a control verb, and co-reference, where multiple men-

⁵https://github.com/jflanigan/jamr/blob/master/docs/Hand_Alignments.md

tions correspond to the same concept.⁶

In contrast, dependency trees do not have nodes with multiple parents. Therefore, when creating a new arc, transition systems for dependency parsing check that the dependent does not already have a head node, preventing the node from having additional parents. To handle reentrancy, which is not uncommon in AMR structures as shown in Table 1, we drop this constraint.

Alignment Another main difference with dependency parsing is that in AMR there is no straightforward mapping between a word in the sentence and a node in the graph: words may generate no nodes, one node or multiple nodes. In addition, the labels at the nodes are often not easily determined by the word in the sentence. For instance *expectation* translates to *expect-01* and *teacher* translates to the two nodes *teach-01* and *person*, connected through an *:ARG0* edge, expressing that a teacher is a person who teaches. A mechanism of concept identification is therefore required to map each token x_i to a subgraph with the correct labels at its nodes and edges: if π is the gold alignment, this should be the subgraph $\overleftarrow{\pi}(i)$ defined in Equation (1). To obtain alignments between the tokens in the sentence and the nodes in the AMR graph of our training data, we run the JAMR aligner.⁷

3 Transition system for AMR Parsing

A **stack** $\sigma = \sigma_n | \dots | \sigma_1 | \sigma_0$ is a list of nodes of the partially constructed AMR graph, with the top element σ_0 at the right. We use the symbol ‘|’ as the concatenation operator. A **buffer** $\beta = \beta_0 | \beta_1 | \dots | \beta_n$ is a list of indices from x , with the first element β_0 at the left, representing the word tokens from the input still to be processed. A **configuration** of our parser is a triple (σ, β, A) , where A is the set of AMR edges that have been constructed up to this point.

In order to introduce the transition actions of our parser we need some additional notation. We use a function a that maps indices from x to AMR graph fragments. For each $i \in [n]$, $a(i)$ is a graph $G_a = (V_a, E_a)$, with single root $\text{root}(G_a)$, representing the semantic contribution of word x_i to the

⁶A valid criticism of AMR is that these two reentrancies are of a completely different type, and should not be collapsed together. Co-reference is a discourse feature, working by extra-semantic mechanisms and able to cross sentence boundaries, which are not crossed in AMR annotation.

⁷<https://github.com/jflanagan/jamr>

AMR for x . As already mentioned, G_a can have a single node representing the concept associated with x_i , or it can have several nodes in case x_i denotes a complex concept, or it can be empty.

The transition **Shift** is used to decide if and what to push on the stack after consuming a token from the buffer. Intuitively, the graph fragment $a(\beta_0)$ obtained from the token β_0 , if not empty, is “merged” with the graph we have constructed so far. We then push onto the stack the node $\text{root}(a(\beta_0))$ for further processing. **L**ARC(ℓ) creates an edge with label ℓ between the top-most node and the second top-most node in the stack, and pops the latter. **R**ARC(ℓ) is the symmetric operation, but does not pop any node from the stack.

Finally, **Reduce** pops the top-most node from the stack, and it also recovers reentrant edges between its sibling nodes, capturing for instance several control verb patterns. To accomplish this, **Reduce** decides whether to create an additional edge between the node being removed and the previously created sibling in the partial graph. With this operation the transition system is able to capture non-projective patterns,⁸ according to the definition given in §2.1, when formed by arcs between nodes that share the same parent. This way of handling control verbs is similar to the *REEN-TRANCE* transition of Wang et al. (2015a).

The choice of popping the dependent in the **L**ARC transition is inspired by **ARCEAGER**, where left-arcs are constructed bottom-up to increase the incrementality of the transition system (Nivre, 2004). This affects our ability to recover some reentrant edges: consider a node u with two parents v and v' , where the arc $v \rightarrow u$ is a left-arc and $v' \rightarrow u$ is any arc. If the first arc to be processed is $v \rightarrow u$, we use **L**ARC that pops u , hence making it impossible to create the second arc $v' \rightarrow u$. Nevertheless, we discovered that this approach works better than a completely unrestricted allowance of reentrancy. The reason is that if we do not remove dependents at all when first attached to a node, the stack becomes larger, and nodes which should be connected end up being distant from each other, and as such, are never connected.

The initial configuration of the system has a \circ node (representing the root) in the stack and the entire sentence in the buffer. The terminal configuration consists of an empty buffer and a stack

⁸In an earlier version of this paper this mechanism was not used, yielding a strictly projective parser.

with only the \circ node. The transitions required to parse the sentence *The boy and the girl* are shown in Table 2, where the first line shows the initial configuration and the last line shows the terminal configuration.

Similarly to the transitions of the ARCEAGER, the above transitions construct edges as soon as the head and the dependent are available in the stack, with the aim of maximizing the parser incrementality. We now show that our greedy transition-based AMR parser is linear-time in n , the length of the input sentence x . We first claim that the output graph has size $\mathcal{O}(n)$. Each token in x is mapped to a constant number of nodes in the graph by **Shift**. Thus the number of nodes is $\mathcal{O}(n)$. Furthermore, each node can have at most three parent nodes, created by transitions **RArc**, **LArc** and **Reduce**, respectively. Thus the number of edges is also $\mathcal{O}(n)$. It is possible to bound the maximum number of transitions required to parse x : the number of **Shift** is bounded by n , and the number of **Reduce**, **LArc** and **RArc** is bounded by the size of the graph, which is $\mathcal{O}(n)$. Since each transition can be carried out in constant time, we conclude that our parser runs in linear time.

4 Training the System

Several components have to be learned: (1) a transition classifier that predicts the next transition given the current configuration, (2) a binary classifier that decides whether or not to create a reentrancy after a **Reduce**, (3) a concept identification step for each **Shift** to compute $a(\beta_0)$, and (4) another classifier to label edges after each **LArc** or **RArc**.

4.1 Oracle

Training our system from data requires an oracle—an algorithm that given a gold-standard AMR graph and a sentence returns transition sequences that maximize the overlap between the gold-standard graph and the graph dictated by the sequence of transitions.

We adopt a shortest stack, static oracle similar to Chen and Manning (2014). Informally, static means that if the actual configuration of the parser has no mistakes, the oracle provides a transition that does not introduce any mistake. Shortest stack means that the oracle prefers transitions where the number of items in the stack is minimized. Given the current configuration (σ, β, A) and the gold-

standard graph $G = (V_g, A_g)$, the oracle is defined as follows, where we test the conditions in the given order and apply the action associated with the first match:

1. if $\exists \ell[(\sigma_0, \ell, \sigma_1) \in A_g]$ then **LArc**(ℓ);
2. if $\exists \ell[(\sigma_1, \ell, \sigma_0) \in A_g]$ then **RArc**(ℓ);
3. if $\neg \exists i, \ell[(\sigma_0, \ell, \beta_i) \in A_g \vee (\beta_i, \ell, \sigma_0) \in A_g]$ then **Reduce**;
4. **Shift** otherwise.

The oracle first checks whether some gold-standard edge can be constructed from the two elements at the top of the stack (conditions 1 and 2). If **LArc** or **RArc** are not possible, the oracle checks whether all possible edges in the gold graph involving σ_0 have already been processed, in which case it chooses **Reduce** (conditions 3). To this end, it suffices to check the buffer, since **LArc** and **RArc** have already been excluded and elements in the stack deeper than position two can no longer be accessed by the parser. If **Reduce** is not possible, **Shift** is chosen.

Besides deciding on the next transition, the oracle also needs the alignments, which we generate with JAMR, in order to know how to map the next token in the sentence to its AMR subgraph $\overleftarrow{\pi}(i)$ defined in (1).

4.2 Transition Classifier

Like all other transition systems of this kind, our transition system has a “controller” that predicts a transition given the current configuration (among **Shift**, **LArc**, **RArc** and **Reduce**). The examples from which we learn this controller are based on features extracted from the oracle transition sequences, where the oracle is applied on the training data.

As a classifier, we use a feed-forward neural network with two hidden layers of 200 tanh units and learning rate set to 0.1, with linear decay-ing. The input to the network consists of the concatenation of embeddings for words, POS tags and Stanford parser dependencies, one-hot vectors for named entities and additional sparse features, extracted from the current configuration of the transition system; this is reported in more details in Table 3. The embeddings for words and POS tags were pre-trained on a large unannotated corpus consisting of the first 1 billion char-

action	stack	buffer	edges
-	[o]	[the,boy,and,the,girl]	{}
Shift	[o]	[boy,and,the,girl]	{}
Shift	[o, boy]	[and,the,girl]	{}
Shift	[o, boy, and]	[the,girl]	{}
LArc	[o, and]	[the,girl]	$\{\langle and, :op1, boy \rangle\} = A_1$
RArc	[o, and]	[the,girl]	$A_1 \cup \{\langle o, :top, and \rangle\} = A_2$
Shift	[o, and]	[girl]	A_2
Shift	[o, and, girl]	[]	A_2
RArc	[o, and, girl]	[]	$A_2 \cup \{\langle and, :op2, girl \rangle\} = A_3$
Reduce	[o, and]	[]	A_3
Reduce	[o]	[]	A_3

Table 2: Parsing steps for the sentence “The boy and the girl.”

acters from Wikipedia.⁹ For lexical information, we also extract the leftmost (in the order of the aligned words) child (c), leftmost parent (p) and leftmost grandchild (cc). Leftmost and rightmost items are common features for transition-based parsers (Zhang and Nivre, 2011; Chen and Manning, 2014) but we found only leftmost to be helpful in our case. All POS tags, dependencies and named entities are generated using Stanford CoreNLP (Manning et al., 2014). The accuracy of this classifier on the development set is 84%.

Similarly, we train a binary classifier for deciding whether or not to create a reentrant edge after a **Reduce**: in this case we use word and POS embeddings for the two nodes being connected and their parent as well as dependency label embeddings for the arcs between them.

4.3 Concept Identification

This routine is called every time the transition classifier decides to do a **Shift**; it is denoted by $a(\cdot)$ in §3. This component could be learned in a supervised manner, but we were not able to improve on a simple heuristic, which works as follows: during training, for each **Shift** decided by the oracle, we store the pair $(\beta_0, \overleftarrow{\pi}(i))$ in a phrase-table. During parsing, the most frequent graph H for the given token is then chosen. In other words, $a(i)$ approximates $\overleftarrow{\pi}(i)$ by means of the graph most frequently seen among all occurrences of token x_i in the training set.

An obvious problem with the phrase-table approach is that it does not generalize to unseen words. In addition, our heuristic relies on the fact that the mappings observed in the data are correct,

⁹<http://mattmahoney.net/dc/enwik9.zip>

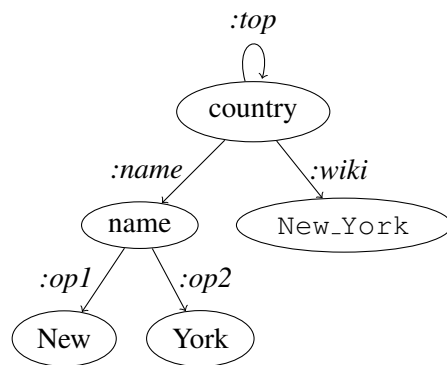


Figure 4: Subgraph for “New York.”

which is not the case when the JAMR-generated alignments contain a mistake. In order to alleviate this problem we observe that there are classes of words such as named entities and numeric quantities that can be disambiguated in a deterministic manner. We therefore implement a set of “hooks” that are triggered by the named entity tag of the next token in the sentence. These hooks override the normal **Shift** mechanism and apply a fixed rule instead. For instance, when we see the token *New York* (the two tokens are collapsed in a single one at preprocessing) we generate the subgraph of Figure 4 and push its root onto the stack. Similar subgraphs are generated for all states, cities, countries and people. We also use hooks for ordinal numbers, percentages, money and dates.

4.4 Edge Labeling

Edge labeling determines the labels for the edges being created. Every time the transition classifier decides to take an **LARC** or **RArc** operation, the edge labeler needs to decide on a label for it. There are more than 100 possible labels such as *:ARG0*,

depth	$d(\sigma_0), d(\sigma_1)$
children	$\#c(\sigma_0), \#c(\sigma_1)$
parents	$\#p(\sigma_0), \#p(\sigma_1)$
lexical	$w(\sigma_0), w(\sigma_1), w(\beta_0), w(\beta_1),$ $w(p(\sigma_0)), w(c(\sigma_0)), w(cc(\sigma_0)),$ $w(p(\sigma_1)), w(c(\sigma_1)), w(cc(\sigma_1))$
POS	$s(\sigma_0), s(\sigma_1), s(\beta_0), s(\beta_1)$
entities	$e(\sigma_0), e(\sigma_1), e(\beta_0), e(\beta_1)$
dependency	$\ell(\sigma_0, \sigma_1), \ell(\sigma_1, \sigma_0),$ $\forall i \in \{0, 1\}: \ell(\sigma_i, \beta_0), \ell(\beta_0, \sigma_i)$ $\forall i \in \{1, 2, 3\}: \ell(\beta_0, \beta_i), \ell(\beta_i, \beta_0)$ $\forall i \in \{1, 2, 3\}: \ell(\sigma_0, \beta_i), \ell(\beta_i, \sigma_0)$

Table 3: Features used in transition classifier. The function d maps a stack element to the depth of the associated graph fragment. The functions $\#c$ and $\#p$ count the number of children and parents, respectively, of a stack element. The function w maps a stack/buffer element to the word embedding for the associated word in the sentence. The function p gives the leftmost (according to the alignment) parent of a stack element, the function c the leftmost child and the function cc the leftmost grandchild. The function s maps a stack/buffer element to the part-of-speech embedding for the associated word. The function e maps a stack/buffer element to its entity. Finally, the function ℓ maps a pair of symbols to the dependency label embedding, according to the edge (or lack of) in the dependency tree for the two words these symbols are mapped to.

:ARG0-of, :ARG1, :location, :time and *:polarity*. We use a feed-forward neural network similar to the one we trained for the transition classifier, with features shown in Table 4. The accuracy of this classifier on the development set is 77%.

We constrain the labels predicted by the neural network in order to satisfy requirements of AMR. For instance, the label *:top* can only be applied when the node from which the edge starts is the special \circ node. Other constraints are used for the *:polarity* label and for edges attaching to numeric quantities.

5 Fine-grained Evaluation

Until now, AMR parsers were evaluated using the Smatch score.¹⁰ Given the candidate graphs and

¹⁰Since Smatch is an approximate randomized algorithm, decimal points in the results vary between different runs and are not reported. This approach was also taken by Wang et al.

name	feature template
depth	$d(\sigma_0), d(\sigma_1)$
children	$\#c(\sigma_0), \#c(\sigma_1)$
parents	$\#p(\sigma_0), \#p(\sigma_1)$
lexical	$w(\sigma_0), w(\sigma_1),$ $w(p(\sigma_0)), w(c(\sigma_0)), w(cc(\sigma_0)),$ $w(p(\sigma_1)), w(c(\sigma_1)), w(cc(\sigma_1))$
POS	$s(\sigma_0), s(\sigma_1)$
entities	$e(\sigma_0), e(\sigma_1)$
dependency	$\ell(\sigma_0, \beta_0), \ell(\beta_0, \sigma_0)$

Table 4: Features used in edge labeling. See Table 3 for a legend of symbols.

the gold graphs in the form of AMR annotations, Smatch first tries to find the best alignments between the variable names for each pair of graphs and it then computes precision, recall and F1 of the concepts and relations. We note that the Smatch score has two flaws: (1) while AMR parsing involves a large number of subtasks, the Smatch score consists of a single number that does not assess the quality of each subtasks separately; (2) the Smatch score weighs different types of errors in a way which is not necessarily useful for solving a specific NLP problem. For example, for a specific problem concept detection might be deemed more important than edge detection, or guessing the wrong sense for a concept might be considered less severe than guessing the wrong verb altogether.

Consider the two parses for the sentence *Silvio Berlusconi gave Lucio Stanca his current role of modernizing Italy’s bureaucracy* in Figure 5. At the top, we show the output of a parser (*Parse 1*) that is not able to deal with named entities. At the bottom, we show the output of a parser (*Parse 2*) which, except for *:name, :op* and *:wiki*, always uses the edge label *:ARG0*. The Smatch scores for the two parses are 56 and 78 respectively. Both parses make obvious mistakes but the three named entity errors in *Parse 1* are considered more important than the six wrong labels in *Parse 2*. However, without further analysis, it is not advisable to conclude that *Parse 2* is better than *Parse 1*. In order to better understand the limitations of the different parsers, find their strengths and gain insight in which downstream tasks they may be helpful, we compute a set of metrics on the test set.

Unlabeled is the Smatch score computed on (2015b) and others.

<pre>(g / give-01 :ARG0 (p3 / silvio :mod (n4 / berlusconi)) :ARG1 (r / role :time (c2 / current) :mod (m / modernize-01 :ARG0 p4 :ARG1 (b / bureaucracy :part-of (c3 / italy))) :poss p4) :ARG2 (p4 / person lucio :mod stanca))</pre>
<pre>(g / give-01 :ARG0 (p3 / person :wiki "Silvio_Berlusconi" :name (n4 / name :op1 "Silvio" :op2 "Berlusconi")) :ARG0 (r / role :ARG0 (c2 / current) :ARG0 (m / modernize-01 :ARG0 p4 :ARG0 (b / bureaucracy :ARG0 (c3 / country :wiki "Italy" :name (n6 / name :op1 "Italy")))) :ARG0 p4) :ARG0 (p4 / person :wiki - :name (n5 / name :op1 "Lucio" :op2 "Stanca"))</pre>

Figure 5: Two parses for the sentence “*Silvio Berlusconi gave Lucio Stanca his current role of modernizing Italy’s bureaucracy.*”

the predicted graphs after removing all edge labels. In this way, we only assess the node labels and the graph topology, which may be enough to benefit several NLP tasks because it identifies basic predicate-argument structure. For instance, we may be interested in knowing whether two events or entities are related to each other, while not being concerned with the precise type of relation holding between them.

No WSD gives a score that does not take into account word sense disambiguation errors. By ignoring the sense specified by the Propbank frame used (e.g., *duck-01* vs *duck-02*) we have a score that does not take into account this additional complexity in the parsing procedure. To compute this score, we simply strip off the suffixes from all Propbank frames and calculate the Smatch score.

Following Sawai et al. (2015), we also evaluate the parsers using the Smatch score on noun phrases only (**NP-only**), by extracting from the AMR dataset all noun phrases that do not include further NPs.

As we previously discussed, reentrancy is a very important characteristic of AMR graphs and it is not trivial to handle. We therefore implement a test for it (**Reentrancy**), where we compute the Smatch score only on reentrant edges.

Concept identification is another critical component of the parsing process and we therefore compute the F-score on the list of predicted concepts (**Concepts**) too. Identifying the correct concepts is fundamental: if a concept is not identified, it will not be possible to retrieve any edge

Metric	First parse	Second parse
Smatch	56	78
Unlabeled	65	100
No WSD	56	78
NP-only	39	86
Reentrancy	69	46
Concepts	56	100
Named Ent.	0	100
Wikification	0	100
Negations	0	0
SRL	69	54

Table 5: Evaluation of the two parses in Figure 5 with the proposed evaluation suite.

involving that concept, with likely significant consequences on accuracy. This metric is therefore quite important to score highly on.

Similarly to our score for concepts, we further compute an F-score on the named entities (**Named Ent.**) and wiki roles for named entities (**Wikification**) that consider edges labeled with *:name* and *:wiki* respectively. These two metrics are strictly related to the concept score. However, since named entity recognition is the focus of dedicated research, we believe it is important to define a metric that specifically assesses this problem. Negation detection is another task which has received some attention. An F-score for this (**Negations**) is also defined, where we find all negated concepts by looking for the *:polarity* role. The reason we can compute a simple F-score instead of using Smatch for these metrics is that there are no variable names involved.

Finally we compute the Smatch score on *:ARG* edges only, in order to have a score for semantic role labeling (**SRL**), which is another extremely important subtask of AMR, as it is based on the identification of predicate-argument structures.

Using this evaluation suite we can evaluate AMRs on a wide range of metrics that can help us find strengths and weakness of each parser, hence speeding up the research in this area. Table 5 reports the scores for the two parses of Figure 5, where we see that *Parse 1* gets a good score for semantic role labeling while *Parse 2* is optimal for named entity recognition. Moreover, we can make additional observations such as that *Parse 2* is optimal with respect to unlabeled score and that *Parse 1* recovers more reentrancies.

Metric	J'14	C'15	J'16	Ours
Smatch	58	63	67	64
Unlabeled	61	69	69	69
No WSD	58	64	68	65
NP-only	47	54	58	55
Reentrancy	38	41	42	41
Concepts	79	80	83	83
Named Ent.	75	75	79	83
Wikification	0	0	75	64
Negations	16	18	45	48
SRL	55	60	60	56

Table 6: Results on test split of LDC2015E86 for JAMR, CAMR and our AMREAGER. J stands for JAMR and C for CAMR (followed by the year of publication). Best systems are in bold.

6 Experiments

We compare our parser¹¹ against two available parsers: JAMR (Flanigan et al., 2014) and CAMR (Wang et al., 2015b; Wang et al., 2015a), using the LDC2015E86 dataset for evaluation. Both parsers are available online¹² and were recently updated for SemEval-2016 Task 8 (Flanigan et al., 2016; Wang et al., 2016). However, CAMR’s SemEval system, which reports a Smatch score of 67, is not publicly available. CAMR has a quadratic worst-case complexity (although linear in practice). In JAMR, the concept identification step is quadratic and the relation identification step is $O(|V|^2 \log |V|)$, with $|V|$ being the set of nodes in the AMR graph.

Table 6 shows the results obtained by the parsers on all metrics previously introduced. On Smatch, our system does not give state-of-the-art results. However, we do obtain the best results for *Unlabeled* and *Concept* and outperform the other parses for *Named Ent.* and *Negations*. Our score of *Reentrancy* is also close the best scoring system, which is particularly relevant given the importance of reentrancies in AMR. The use of the **Reduce** transition, which targets reentrancies caused by control verbs, is critical in order to achieve this result.

The relatively high results we obtain for the un-

¹¹Our parser is available at <https://github.com/mdtux89/amr-eager>, the evaluation suite at <https://github.com/mdtux89/amr-evaluation> and a demo at <http://cohort.inf.ed.ac.uk/amreager.html>

¹²JAMR: <https://github.com/jflanigan/jamr>, CAMR: <https://github.com/c-amr/camr>.

labeled case suggests that our parser has difficulty in labeling the arcs. Our score for concept identification, which is on par with the best result from the other parsers, demonstrates that there is a relatively low level of token ambiguity. State-of-the-art results for this problem can be obtained by choosing the most frequent subgraph for a given token based on a phrase-table constructed from JAMR alignments on the training data. The scores for named entities and wikification are heavily dependent on the hooks mentioned in §4.3, which in turn relies on the named entity recognizer to make the correct predictions. In order to alleviate the problem of wrong automatic alignments with respect to polarity and better detect negation, we performed a post-processing step on the aligner output where we align the AMR constant - (minus) with words bearing negative polarity such as *not*, *illegitimate* and *asymmetry*.

Our experiments demonstrate that there is no parser for AMR yet that conclusively does better than all other parsers on all metrics. Advantages of our parser are the worst-case linear complexity and the fact that is possible to perform incremental AMR parsing, which is both helpful for real-time applications and to investigate how meaning of English sentences can be built incrementally left-to-right.

7 Conclusion

We presented a transition system that builds AMR graphs in linear time by processing the sentences left-to-right. The system is trained with feed-forward neural networks. The parser demonstrates that it is possible to perform AMR parsing using techniques inspired by techniques from dependency parsing.

We also noted that it is less informative to evaluate the entire parsing process with Smatch than to use a collection of metrics aimed at evaluating the various subproblems in the parsing process. We further showed that our left-to-right transition system is competitive with publicly available state-of-the-art parsers. Although we do not outperform the best baseline in terms of Smatch score, we show on par or better results for several of the metrics proposed. We hope that moving away from a single-metric evaluation will further speed up progress in AMR parsing.

Acknowledgments

The authors would like to thank Sameer Bansal, Jeff Flanigan, SORCHA Gilroy, Adam Lopez, Nikos Papasrantopoulos, Nathan Schneider, Mark Steedman, Sam Thomson, Clara Vania and Chuan Wang for their help and comments. This research was supported by a grant from Bloomberg and by the H2020 project SUMMA, under grant agreement 688139.

References

- Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (UCCA). In *Proceedings of ACL*.
- Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. *Proceedings of EMNLP*.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. *Proceedings of Linguistic Annotation Workshop*.
- Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. *Proceedings of ACL*.
- Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*.
- Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, pages 459–464.
- Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. *Proceedings of ACL*.
- Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. CMU at SemEval-2016 task 8: Graph-based AMR parsing with infinite ramp loss. *Proceedings of SemEval*, pages 1202–1206.
- James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. *Proceedings of ACL*.
- Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, pages 559–570.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.
- Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together. ACL*.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics, Volume 34, Number 4, December 2008*.
- Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. *Proceedings of CoNLL*.
- Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Using syntax-based machine translation to parse english into abstract meaning representation. *arXiv preprint arXiv:1504.06665*.
- Sudh Rao, Yogarshi Vyas, Hal Daume III, and Philip Resnik. 2015. Parser for abstract meaning representation using learning to search. *arXiv:1510.07586*.
- Corentin Ribeyre, Éric Villemonte de La Clergerie, and Djamé Seddah. 2015. Because syntax does matter: Improving predicate-argument structures parsing using syntactic features. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Kenji Sagae and Jun’ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. *Proceedings of COLING*.
- Yuichiro Sawai, Hiroyuki Shindo, and Yuji Matsumoto. 2015. Semantic structure analysis of noun phrases using abstract meaning representation. *Proceedings of ACL*.
- Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.
- Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.
- Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for english, french, german, spanish and japanese and a new AMR-annotated corpus. *Proceedings of NAACL-HLT*.
- Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. *Proceedings of ACL*.
- Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. *Proceedings of NAACL*.

Chuan Wang, Sameer Pradhan, Nianwen Xue, Xiaoman Pan, and Heng Ji. 2016. CAMR at SemEval-2016 task 8: An extended transition-based AMR parser. *Proceedings of SemEval*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. *Proceedings of ACL*.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Integrated Learning of Dialog Strategies and Semantic Parsing

Aishwarya Padmakumar

Jesse Thomason

Raymond J. Mooney

Department of Computer Science, University of Texas at Austin

{aish, jesse, mooney}@cs.utexas.edu

Abstract

Natural language understanding and dialog management are two integral components of interactive dialog systems. Previous research has used machine learning techniques to individually optimize these components, with different forms of direct and indirect supervision. We present an approach to integrate the learning of *both* a dialog strategy using reinforcement learning, and a semantic parser for robust natural language understanding, using only natural dialog interaction for supervision. Experimental results on a simulated task of robot instruction demonstrate that joint learning of both components improves dialog performance over learning either of these components alone.

1 Introduction

Natural language understanding and dialog management are two integral components of a dialog system. Current research typically deals with optimizing only one of these components. We present an approach to integrate the learning of *both* a dialog strategy using reinforcement learning, and a semantic parser for robust natural language understanding, using only natural dialog interaction for supervision.

Research in dialog systems has primarily been focused on the problems of accurate dialog state tracking and learning a policy for the dialog system to respond appropriately in various scenarios. Dialogs are typically modeled using Partially Observable Markov Decision Processes (POMDPs), and various reinforcement learning algorithms have been proposed and evaluated for the task of learning optimal policies over these representations to accomplish user goals using as short and

natural a dialog as possible (Gašić and Young, 2014; Pietquin et al., 2011; Young et al., 2013). However, such systems typically assume a fixed language understanding component that is available a priori.

Semantic parsing is the task of mapping natural language to a formal meaning representation. It has the potential to allow for more robust mapping of free-form natural language to a representation that can be used to interpret user intentions and track dialog state. This is done by leveraging the compositionality of meaning inherent in language. Prior work has shown that a semantic parser, incrementally updated from conversations, is helpful in dialogs for communicating commands to a mobile robot (Thomason et al., 2015). We show that incremental learning of a POMDP-based dialog policy allows for further improvement in dialog success.

A major challenge with combining the above parser and dialog policy learning techniques is that reinforcement learning (RL) algorithms assume that the dialog agent is operating in a *stationary* environment. This assumption is violated when the parser is updated between conversations. For example, the improved semantic parser may be able to extract more information from a response to a question, which the old parser could not parse. So the RL algorithm may have earlier assumed that asking such a question is not useful, but this is not the case with the updated parser. Our results show that this effect can be mitigated if we break the allowed budget of training dialogs into batches, updating both parser and policy after each batch. As the next training batch gets collected using the updated parser, the policy can be updated using this experience to adapt better to it. We demonstrate, using crowd-sourced results with a simulated robot, that by integrating learning of *both* a dialog manager *and* a semantic parser in

this manner, task success is improved over cases where the components are trained individually.

2 Related Work

Prior work has used dialog to facilitate robot task learning, e.g. She et al. (2014), but does not account for uncertainty or dynamic changes to the language understanding module when developing a system policy. Some works use a POMDP model and common-sense knowledge (Zhang and Stone, 2015) or generate clarification questions in a probabilistic manner (Tellex et al., 2014), but these too assume that a fixed and well-trained natural language understanding component is available a-priori. Kollar et al. (2013) use a probabilistic parsing and grounding model to understand natural language instructions and extend their knowledge base by asking questions. However, unlike this work, they do not use semantic parsing to leverage the compositionality of language, and also use a fixed hand-coded policy for dialog.

There has been considerable work in semantic parsing using both direct supervision in the form of annotated meaning representations (Wong and Mooney, 2007; Kwiatkowski et al., 2013; Berant et al., 2013) and indirect signals from downstream tasks (Artzi and Zettlemoyer, 2011; Artzi and Zettlemoyer, 2013; Thomason et al., 2015). Artzi and Zettlemoyer (2011) use clarification dialogs to train semantic parsers for an airline reservation system without explicit annotation of meaning representations. More related to our work is that of Thomason et al. (2015), who incorporated this general approach into a system for instructing a mobile robot; however, they use a simple model of dialog state and a fixed, hand-coded dialog policy. We show that learning a dialog policy in addition to this, is more beneficial than only parser learning. We also use a richer state representation that incorporates multiple hypotheses from the semantic parser.

There has also been considerable work in goal-directed dialog systems in domains such as information provision (Young et al., 2013). These systems model dialog as a POMDP and focus on either the problem of tracking belief state accurately over large state spaces (Young et al., 2010; Thomson and Young, 2010; Mrkšić et al., 2015; El Asri et al., 2016) or efficiently learning a dialog policy over this state space (Gašić and Young, 2014; Pietquin et al., 2011; Png et al., 2012). However,

these systems typically assume a fixed natural language understanding component. In this work, we combine language learning with principled dialog strategy learning.

More recently, there has been work on modeling various components of a dialog system using neural networks (Mrkšić et al., 2015; Wen et al., 2015). There have also been some end-to-end neural network systems that simultaneously learn dialog policy and language comprehension for goal directed dialog (Wen et al., 2016; Williams and Zweig, 2016; Bordes and Weston, 2016), but they do not use a fully compositional semantic parser. Williams and Zweig (2016) use a very simple keyword-spotting based technique for processing input user utterances, which is unlikely to be able to handle out-of-vocabulary expressions for entities. Bordes and Weston (2016) explicitly attempt to handle out-of-vocabulary utterances in a neural dialog system but do not demonstrate much success. We expect that in a domain such as ours where out-of-vocabulary utterances are fairly likely, for example, in different forms of address for a person, a semantic parser that can be incrementally updated from a small number of interactions is likely to perform better. However, an empirical comparison of the two in domains where compositional language understanding is expected to be beneficial, is an interesting direction of future work.

3 Background - Partially Observable Markov Decision Process (POMDP)

A Partially Observable Markov Decision Process (POMDP) is a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$, where \mathbb{S} is a set of states, \mathbb{A} is a set of actions, \mathbb{T} is a transition function, \mathbb{R} is a reward function, \mathbb{O} is a set of observations, \mathbb{Z} is an observation function, γ is a discount factor and b_0 is an initial belief state (Kaelbling et al., 1998). These are defined as follows.

At any instant of time t , the agent is in a state $s_t \in \mathbb{S}$. This state is hidden from the agent and only a noisy observation $o_t \in \mathbb{O}$ of s_t is provided to it. The agent maintains a belief state b_t which is a distribution over all possible states it could be in at time t , where $b_t(s_i)$ gives the probability of being in state s_i at time t . Based on b_t , the agent chooses to take an action $a_t \in \mathbb{A}$ according to a policy π , commonly represented as a probability distribution over actions where $\pi(a_t|b_t)$ is the

probability of taking action a_t when the agent is in belief state b_t . On taking action a_t , the agent is given a real-valued reward r_t , transitions to a state s_{t+1} , and receives a noisy observation o_{t+1} of s_{t+1} .

State transitions occur according to the probability distribution $P(s_{t+1}|s_t, a_t) = \mathbb{T}(s_t, a_t, s_{t+1})$, observations are related to the states by the probability distribution $P(o_t|s_t, a_{t-1}) = \mathbb{Z}(o_t, s_t, a_{t-1})$ and rewards obtained follow the distribution $P(r_t|s_t, a_t) = \mathbb{R}(s_t, a_t, s_{t+1})$.

The objective is to identify a policy π that is optimal in the sense that it maximizes the expected long term discounted reward, called return, given by

$$g = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \right]$$

While there exist both exact and approximate methods for solving POMDPs, these do not usually scale well to the state spaces commonly used in dialog domains. This has led to the development of approximate representations that exploit domain-specific properties of dialog tasks to allow tractable estimation of the belief state and policy optimization (Young et al., 2013).

4 Background - Q-Learning using Kalman Temporal Differences

The quality of a policy π can be estimated using the action value function

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$$

The optimal policy satisfies the Bellman equation,

$$Q^*(s, a) = \mathbb{E}_{s'} \left[\mathbb{R}(s, a, s') + \gamma \max_{a' \in \mathbb{A}} Q^*(s', a') \right]$$

When the state space is very large or continuous, Q^π cannot be computed for each state (or belief state) individually and is hence assumed to be a function with parameters θ over some features that represent the state. When the transition or reward dynamics are not constant (*non-stationary* problem), a suitable approximation is the Kalman Temporal Differences framework (Geist and Pietquin, 2010). This casts the function approximation as a filtering problem and solves it using Kalman filtering. The specialization for learning the optimal action value function is called the KTD-Q algorithm.

Filtering problems estimate hidden quantities X from related observations Y, modeling X and Y as random variables. When estimating action values, X corresponds to the function parameters, θ and the observations, Y, are the estimated returns, $r_t + \gamma \max_a \hat{Q}_{\theta_t}(s_{t+1}, a)$. Random noise is added to both of these to allow for parameters to change over time. The update rules are derived from Kalman Filtering Theory and not included here for the sake of brevity.

5 Dialog System

Our system initiates the dialog by requesting the user for a command. The user can command the system to perform two actions: navigation and delivery. Navigation has a single parameter for the destination. For example “*go to Alice’s office*” would be a possible way to command the robot to perform a navigation command, whose location is a room that is the office of a person `alice`. Delivery has two parameters: the item to be delivered and the person to receive it. For example, “*bring Alice a hamburger*” would be a possible way to specify a delivery command whose patient is an item `hamburger` and recipient is a person `alice`.

The robot makes an initial guess of the desired action from the user’s response, and then may ask clarification questions in case of insufficient understanding. At each step, it can respond with one of four dialog acts: asking the user to repeat their command, confirming a command or an argument value, requesting a specific argument of a command, and executing an action (thereby ending the dialog). A sample dialog is shown in Table 1.

Table 1: An sample clarification dialog

ROBOT	How can I help?
USER	bring the coffee to dr morgan
ROBOT	Sorry I couldn’t understand that. Could you reword your original request?
USER	take the coffee to dr morgan
ROBOT	Sorry I couldn’t understand that. Could you reword your original request?
USER	give the coffee cup to dr morgan
ROBOT	You want me to bring coffee for George?
USER	no mallory morgan
ROBOT	Whom would you like me to bring something to?
USER	mallory morgan
ROBOT	What would you like me to bring?
USER	coffee
ROBOT	I brought coffee to Mallory. Was this the correct action?
USER	yes

Given a user response, a semantic parser is used to convert utterances to logical forms which are then grounded using world knowledge (e.g. resolving the representation of “*Alice’s office*” to a particular room). These grounded logical forms are then used to update the belief state maintained by the system. The next step to be taken by the system, given the belief state, is then chosen based on the current dialog policy. Once the conversation is complete, the parser and policy can be updated appropriately. These steps are outlined in greater detail in sections 5.1 and 5.2.

The dialog is considered a success if the final action taken is correct and a failure otherwise. The user also has the ability to prematurely end the dialog, and any conversation terminated in this manner is also considered a failure.

5.1 Semantic Parser Learning

Semantic parsing maps a natural language sentence such as “*Go to Alice’s office*” to a logical form expressed in λ -calculus such as:

$$\text{walk}(\text{the}(\lambda x.\text{office}(x) \wedge \text{possess}(\text{alice}, x) \wedge \text{person}(\text{alice}))) \quad (1)$$

Grounding against real-world knowledge, this will identify a room, say room 3512, which is an office that is owned by *alice*.

This formalism reduces the number of lexical entries the system needs to learn by exploiting compositional reasoning over language. For example, if the system learns that “*Alice Ashcraft*” and “*Alice*” both refer to the entity *alice*, no further lexical entries are required to resolve “*Go to Alice Ashcraft’s office*” to the same semantic form (1).

In our system, semantic parsing is performed using probabilistic CKY-parsing with a Combinatory Categorical Grammar (CCG) and meanings associated with lexical entries. Perceptron-style updates to parameter values, that minimize the log-likelihood of the training data, are used during training to weight parses to speed search and give confidence scores in parse hypotheses (Zettlemoyer and Collins, 2005).

The parser is trained using paired sentences and logical forms. A small supervised training set is used to initialize the parser. Training continues using pairs obtained through weak supervision collected from user dialogs (Thomason et al., 2015).

We use two such types of training pairs. The first consist of responses that are likely to correspond to the complete action, and the logical form induced by the action executed by the robot at the end of the dialog. Such responses are expected from the initial prompt to the user and questions that ask the user to repeat the command. We obtain multiple semantic parses for these responses, and parses that correspond to a complete command, and ground to the action finally taken by the robot, are paired with the response to form one set of training pairs. For example, from the conversation in Table 1, such training examples would be generated by pairing the responses “*bring the coffee to dr morgan*”, “*take the coffee to dr morgan*” and “*give the coffee cup to dr morgan*” with the semantic form `bring(mallory, coffee)`.

The second set of training pairs is obtained from the arguments of the action, such as the patient or location involved. This consists of responses to requests for specific arguments. Again, we consider multiple semantic parses for these responses, and select those that are of the correct syntactic form for a single argument value, and which ground to the corresponding argument value in the final action, to be paired with the response. For example, from the conversation in Table 1, such training examples would be generated by pairing the response “*mallory morgan*” with the semantic form `mallory`, and the response “*coffee*” with the semantic form `coffee`. These paired responses and semantic forms can then be used to retrain the parser between conversations.

This weak supervision may be somewhat noisy because it assumes that the form of the user’s response matches the expected response type for the question. However, this is unlikely to generate spurious training examples, because we additionally place constraints on the syntax of the response. For example, if we receive “*Go to Bob’s office*” as a response when we expect an argument value, since the response is an imperative sentence, not a noun phrase such as “*Bob’s office*”, no training example would be generated from it. Prior experimental results (Artzi and Zettlemoyer, 2011; Thomason et al., 2015) suggest that learning using such weak (potentially noisy) supervision from clarification dialogs is effective at improving semantic parsers.

5.2 Dialog Strategy Learning

We use a POMDP to model dialog and learn a policy (Young et al., 2013), adapting the Hidden Information State model (HIS) (Young et al., 2010) to track the belief state as the dialog progresses. The key idea behind this approach is to group states into equivalence classes called partitions, and maintain a probability for each partition instead of each state. States within a partition are those that are indistinguishable to the system given the current dialog.

More concretely, our belief state can be factored into two main components. The first is the action (such as navigation and delivery) and argument values of the goal (such as the patient or location) which the user is trying to convey, $\mathbf{g} = \{g_a, g_{PAT}, g_{RCP}, g_{LOC}\}$. Goal parameters are represented in terms of semantic roles - *patient* (g_{PAT}), *recipient* (g_{RCP}) and *location* (g_{LOC}), to allow them to generalize across different actions. The second component contains information from the most recent user utterance, $\mathbf{u} = \{u_t, u_a, u_{PAT}, u_{RCP}, u_{LOC}\}$. Here, u_t is the type of the utterance – affirmation, denial, providing information about a complete action, or providing information about a specific argument. The components u_a , u_{PAT} , u_{RCP} and u_{LOC} respectively refer to the action, *patient*, *recipient* and *location* mentioned in the most recent user utterance, any of which can be `null`. This representation allows the method to be applicable to any action that can be expressed using up to 3 arguments.

After every user response, a beam of possible choices for \mathbf{u} can be obtained by grounding the beam of top-ranked parses from the semantic parser. Semantic type-checking is used to disallow violations such as `alice` serving as the location argument of a navigation. However, there are a large number of possible values for \mathbf{g} and we use the idea of partitions (Young et al., 2010) to track their probabilities in a tractable manner. A partition is a set of possible goals $\mathbf{g}^{(i)}$ which are equally probable given the conversation so far. The probability of a partition is the sum of probabilities of all goals in the partition. Initially, all goals are in a single partition of probability 1.

When an utterance hypothesis \mathbf{u} is obtained, every partition currently maintained is split if needed into partitions that are either completely consistent or inconsistent with \mathbf{u} . For example, if a partition p has goals containing both navigation and deliv-

ery actions, and \mathbf{u} specifies a delivery action, p will have to be split into one partition p_1 with all the navigation goals and another partition p_2 with all the delivery goals. The probability mass of p is divided between p_1 and p_2 in proportion to their sizes, to maintain the invariant that the probability of a partition is the sum of the probabilities of the goals contained in it. Then, given the previous system action \mathbf{m} , The belief $b(p, \mathbf{u})$ is calculated as in the HIS model as follows

$$b(p, \mathbf{u}) = k * P(\mathbf{u}) * T(\mathbf{m}, \mathbf{u}) * M(\mathbf{u}, \mathbf{m}, p) * b(p)$$

Here, $P(\mathbf{u})$ is the probability of the utterance hypothesis \mathbf{u} given the user response, which is obtained from the semantic parser. $T(\mathbf{m}, \mathbf{u})$ is the probability that the type of the utterance hypothesis \mathbf{u}_t is compatible with the previous system action \mathbf{m} , for example, if the system asks for the confirmation of a goal, the expected type of response is either affirmation or denial. This is determined by system parameters. $M(\mathbf{u}, \mathbf{m}, p)$ is a 0-1 value indicating whether the action and argument values mentioned in the utterance, system action, and partition agree with each other (an example of where they do not is an utterance mentioning an action not present in any goal in the partition) and $b(p)$ is the belief of partition p before the update, obtained by marginalizing out \mathbf{u} from $b(p, \mathbf{u})$. k is a normalization constant that allows the expression to become a valid probability distribution. We also track the number of dialog turns so far.

The belief state is a distribution over all possible hypotheses given the conversation so far. The HIS model allows tracking probabilities of the potentially large number of hypotheses. However, it is difficult to learn a policy over this large a state space in a reasonable number of dialogs. Thus, we learn a dialog policy over a summary state as in previous work (Young et al., 2010; Gašić and Young, 2014). Table 2 contains the features used to learn the policy. Also, the policy is learned over abstract dialog acts (ask user to rephrase the entire goal, ask for a specific parameter, confirm a full/partial goal, execute a goal), which are converted to a system response by using parameters from the most likely hypothesis.

It is important to note that while only the top two hypotheses are used by the policy to choose the next action, it is useful to maintain the belief of all hypotheses because a hypothesis that is initially of low probability may become the most probable after additional turns of dialog.

Probability of top hypothesis
Probability of second hypothesis
Number of goals allowed by the partition in the top hypothesis
Number of parameters of the partition in the top hypothesis, required by its action, that are uncertain (set to the maximum value if there is more than one possible action)
Number of dialog turns used so far
Do the top and second hypothesis use the same partition (0-1)
Type of last user utterance
Action of the partition in the top hypothesis, or <i>null</i> if this is not unique

Table 2: Features used in summary space

The choice of policy learning algorithm is important because learning POMDP policies is challenging and dialog applications exhibit properties not often encountered in other reinforcement learning applications (Daubigney et al., 2012). We use KTD-Q (Kalman Temporal Difference Q-learning (Geist and Pietquin, 2010)) to learn the dialog policy as it was designed to satisfy some of these properties and tested in a dialog system with simulated users (Pietquin et al., 2011). The properties we wished to be satisfied by the algorithm were the following:

- Low sample complexity in order to learn from limited user interaction.
- An off-policy algorithm to enable the use of existing dialog corpora to bootstrap the system, and crowdsourcing platforms such as Amazon Mechanical Turk during training and evaluation.
- A model-free rather than a model-based algorithm because it is difficult to design a good transition and observation model for this problem (Daubigney et al., 2012).
- Robustness to non-stationarity because the underlying language understanding component changes with time (Section 5.1), which is likely to change state transitions.

To learn the policy, we provided a high positive reward for correct completion of the task and a high negative reward when the robot chose to execute an incorrect action, or if the user terminated the dialog before the robot was confident about taking an action. The system was also given a per-turn reward of -1 to encourage shorter dialogs.

6 Experimental Evaluation

The learning methods described above were applied to improve an initial dialog system using weak supervision from dialog interaction with real users. The dialog system was initialized using data from the conversation logs of Thomason et al. (2015), which also consist of interactions between a human user and a robot to which a high-level command must be communicated, and which asks clarifying questions when attempting to understand the dialog.

6.1 Initialization

The semantic parser was initialized using a small seed lexicon and trained on a small set of supervised examples constructed using templates for commands gathered from the conversation logs. While the parser can be used even if initialized using only a handful of hand-coded training examples, the increased robustness obtained by training on templated sentences results in less frustrating interaction during initial dialogs.

The RL component was first initialized with a Q -function approximation of the hand-coded policy of Thomason et al. (2015). The hand-coded policy was encoded in the form of if-then rules and had to be mapped to a Q -function appropriate for the KTD-Q algorithm, which assumes the Q -function is a probability distribution with a mean that is a linear function of the feature space. We obtain a set of “training points” for these linear weights by densely sampling the feature space. The hand coded policy is then used to identify the correct action for each of these feature vectors. The target for a training point is a high positive Q value when combined with the correct action and a 0 value when combined with any incorrect action. The weights were then initialized using linear regression over these examples. Finally, we trained the system on the above mentioned conversation logs, improving both the initial POMDP dialog policy and the semantic parser.

The simplest alternative to such an initialization would be to initialize the policy at random, but this would lead to a large number of frustrating dialogs before the system learns a reasonable policy. This can be avoided by training with a simulated user agent. However, such agents are not always realistic and their design requires parameters to be set ideally from existing conversation logs. However, since we use an off-policy algorithm, it is easier to

train it directly from conversation logs, rather than develop a sufficiently realistic simulated agent.

Since the KTD-Q algorithm is off-policy, it can be trained using tuples containing the belief state, action taken, next belief state, and reward obtained from these logs. We update the policy using such tuples both in the initial training phase from existing conversation logs, and when updating the policy after collecting batches of conversations in our experiments.

6.2 Platform and setup

Our experiments were done through Mechanical Turk as in previous work (Thomason et al., 2015; Wen et al., 2016). During the training phase, each user interacted with one of four dialog agents (described in section 6.3), selected uniformly at random. Users were not told of the presence of multiple agents and were not aware of which agent they were interacting with. They were given a prompt for either a navigation or delivery task and were asked to have a conversation with the agent to accomplish the given task. No restrictions were placed on the language they could employ. We use visual prompts for the tasks to avoid linguistic priming (e.g. a picture of a hamburger instead of the word “hamburger”). Before users could begin the task, we used a validation step to ensure they were sufficiently fluent in English and understood the objectives of the task. Training dialogs were acquired in 4 batches of 50 dialogs each across all agents. After each batch, agents were updated as described in section 6.3.

A final set of 100 test conversations were then conducted between Mechanical Turk users and the trained agents. These test tasks were novel in comparison to the training data in that although they used the same set of possible actions and argument values, the same combination of action and argument values had not been seen at training time. For example, if one of the test tasks involved delivery of a `hamburger` to `alice`, then there may have been tasks in the training set to deliver a `hamburger` to other people and there may have been tasks to deliver other items to `alice`, but there was no task that involved delivery of a `hamburger` to `alice` specifically.

6.3 Dialog agents

We compared four dialog agents. The first agent performed only parser learning (described in Section 5.1). Its dialog policy was always kept to be a

hand coded dialog policy similar to that of Thomason et al. (2015). This was the same hand-coded policy used to initialize the weights of the KTD-Q algorithm. Its parser was incrementally updated after each training batch. This agent is similar to the system used by Thomason et al. (2015) except that it uses the same state space as our other agents, to ensure that any differences in performance are not due to access to less information. Further, while Thomason et al. (2015) use only the top hypothesis from the parser to update the belief state, our agent uses a beam of parses, again to be more comparable to our other agents. In supplementary material, we also include an experiment which demonstrates that using multiple hypotheses from the semantic parser is more beneficial than using only a single one.

The second agent performed only dialog strategy learning. Its parser was always kept to be the initial parser that all agents started out with. Its policy was incrementally updated after each training batch using the KTD-Q algorithm. The third agent performed both parser and dialog learning; but instead of incrementally updating the parser and policy after each batch, they were trained at the end of the training phase using dialogs across all batches. This would not allow the dialog manager to see updated versions of the parser in batches after the first and adapt the policy towards the improving parser. We refer to this as *full* learning of parser and dialog policy. The fourth agent also performed both parser and dialog learning. Its parser and policy were updated incrementally after each training batch. Thus for the next training batch, the changes due to the improvement in the parser from the previous batch could, in theory, be demonstrated in the dialogs and hence contribute towards updating the policy in a manner consistent with it. We refer to this as *batchwise* learning of parser and dialog policy.

We did not include a system that performs no learning on either the parser or policy because it was shown by Thomason et al. (2015) that parser learning combined with a simple hand-coded policy outperforms this. We also did not attempt to update both parser and policy after each dialog because this forces all dialogs to be conducted in sequence, which does not allow us to fully leverage crowdsourcing platforms such as Mechanical Turk.

6.4 Experiment hypothesis

We hypothesized that the agent performing *batchwise* parser and policy learning would outperform the agents performing only parser or only dialog learning as we expect that improving both components is more beneficial. However, we did not necessarily expect the same result from *full* parser and dialog learning because it did not provide any chance to allow updates to propagate even indirectly from one component to another, exposing the RL algorithm to a more *non-stationary* environment. Hence, we also expected *batchwise* learning to outperform *full* learning.

6.5 Results and Discussion

The agents were evaluated on the test set using the following objective performance metrics: the fraction of successful dialogs (see 5) and the length of successful dialogs. We also included a survey at the end of the task asking users to rate on a 1–5 scale whether the robot understood them, and whether they felt the robot asked sensible questions.

Learning involved	% successful dialogs	Avg dialog length	Robot understood	Sensible questions
Parser	75	12.43	2.93	2.79
Dialog	59	11.73	2.55	2.91
Parser & Dialog - <i>full</i>	72	12.76	2.79	3.28
Parser & Dialog - <i>batchwise</i>	78	10.61	3.30	3.17

Table 3: Performance metrics for dialog agents tested. Differences in dialog success and subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$.

Table 3 gives the agents’ performance on these metrics. All differences in dialog success and the subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$. In dialog length, the improvement of the *batchwise* learning agent over the agents performing only parser or only dialog learning are statistically significant.

As expected, the agent performing *batchwise* parser and dialog learning outperforms the agents performing only parser or only dialog learning, in the latter case by a large margin. We believe the agent performing only parser learning performs much better than the agent performing only dialog

learning due to the relatively high sample complexity of reinforcement learning algorithms in general, especially in the partially observable setting. In contrast, the parser changes considerably even from a small number of examples. Also, we observe that *full* learning of both components does not in fact outperform only parser learning. We believe this is because the distribution of hypotheses obtained using the initial parser at training time is substantially different from that obtained using the updated parser at test time. We believe that *batchwise* training mitigates this problem because the distribution of hypotheses changes after each batch of training and the policy when updated at these points can adapt to some of these changes. The optimal size of the batch is a question for further experimentation. Using a larger batch is less likely to overfit updates to a single example but breaking the total budget of training dialogs into more batches allows the RL algorithm to see less drastic changes in the distribution of hypotheses from the parser.

We include an experiment in the supplementary material that quantifies the accuracy improvement of the parsers after training from dialogs. It is more difficult to quantitatively compare the policies before and after learning. Qualitatively, one of the noticeable differences is that the system tends to confirm or act upon lower probability hypotheses than is recommended by the initial hand-coded policy. This is possibly because as the parser improves, its top hypotheses are more likely to be correct, even if they are associated with a lower confidence score from the parser. A demonstration of this can be seen in tables 4 and 5. The learned policy results in a shorter dialog in the same situation because it allows the agent to act upon a hypothesis of lower probability. Also, the learned policy is stochastic, which is very helpful when the agent is not able to understand the user at all. For example, if the agent is unable to parse any of the initial instructions from the user, under a hand-coded policy, as its state has not changed, it would continue to repeat the question it had asked earlier, which prevents it from making any progress. However, in a stochastic policy, other more specific questions are likely to be substituted in between, and responses to these may allow the agent to make progress, which increases dialog success (table 6).

7 Conclusions

In this work, we have demonstrated that continuous dialog strategy learning and semantic parser learning can be successfully combined in a dialog system to enable an agent to better understand commands provided in natural language. Both the semantic parser and the dialog strategy can be automatically improved simultaneously using weak feedback provided during interaction with users rather than manually-labeled or artificially constructed training data. Ongoing parser learning could have confused the RL dialog learner by altering the underlying language understanding system while it was searching for an effective dialog policy. However, our results show that by using an appropriate RL algorithm and batchwise training regimen, this potential difficulty can be avoided, and both language understanding and dialog management can be improved simultaneously.

Acknowledgements

We would like to thank the members of the UT Austin BWI group for several insightful discussions, and Subhashini Venugopalan for her help with revising the paper draft. This research was supported by NSF EAGER (IIS-1548567) and NRI (IIS-1637736) grants.

Table 4: The robot following the learned policy takes the action despite the lower confidence, leading to a shorter dialog.

ROBOT How can I help?
USER bring the hamburger to peggy
ROBOT What would you like me to bring?
USER the hamburger
The robot brought a hamburger to Peggy.
ROBOT Was this the correct action?
USER yes

Table 5: The robot following the hand-coded policy confirms the action before taking it, leading to a longer dialog.

ROBOT How can I help?
USER bring the hamburger to peggy
ROBOT What would you like me to bring?
USER the hamburger
ROBOT You want me to bring hamburger for Peggy?
USER yes
The robot brought a hamburger to Peggy.
ROBOT Was this the correct action?
USER yes

Table 6: An example where a stochastic policy helps when the robot is unable to understand a full command.

ROBOT How can I help?
USER bring evelyn a coffee
ROBOT Sorry I couldn't understand that. Could you reword your original request?
USER bring evelyn a cup of coffee
ROBOT You want me to bring coffee for someone?
USER yes
ROBOT Whom would you like me to bring coffee to?
USER evelyn
ROBOT Whom would you like me to bring coffee to?
USER eve
ROBOT I brought coffee to Eve. Was this the correct action?
USER yes

References

- Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping Semantic Parsers from Conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432.
- Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.
- Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization. *Journal of Selected Topics in Signal Processing*, 6(8):891–902.
- Layla El Asri, Romain Laroche, and Olivier Pietquin. 2016. Compact and Interpretable Dialogue State Representation with Genetic Sparse Distributed Memory. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*.
- Milica Gašić and Steve Young. 2014. Gaussian Processes for POMDP-Based Dialogue Manager Optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.
- Matthieu Geist and Olivier Pietquin. 2010. Kalman Temporal Differences. *Journal of Artificial Intelligence Research*, 39(1):483–532.
- Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and Acting in

- Partially Observable Stochastic Domains. *Artificial intelligence*.
- Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso. 2013. Learning Environmental Knowledge from Task-Based Human-Robot Dialog. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4309.
- Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-Domain Dialog State Tracking Using Recurrent Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient Batch Reinforcement Learning for Dialogue Management Optimization. *ACM Transactions on Audio, Speech, and Language Processing*, 7(3):7:1–7:21.
- Shaowei Png, Joelle Pineau, and Brahim Chaib-Draa. 2012. Building Adaptive Dialogue Systems Via Bayes-Adaptive POMDPs. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):917–927.
- Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014. Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97.
- Stefanie Tellex, Ross A. Knepper, Adrian Li, Nicholas Roy, and Daniela Rus. 2014. Asking for Help Using Inverse Semantics. In *Proceedings of the 2016 Robotics: Science and Systems Conference (RSS)*.
- Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to Interpret Natural Language Commands through Human-Robot Dialog. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1923–1929.
- Blaise Thomson and Steve Young. 2010. Bayesian Update of Dialogue State: A POMDP framework for Spoken Dialogue Systems. *Computer Speech and Language*, 24(4):562–588.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing (EMNLP)*.
- Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.
- Yuk Wah Wong and Raymond J. Mooney. 2007. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 960–967.
- Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State Model: A Practical Framework for POMDP-based Spoken Dialogue Management. *Computer Speech and Language*, 24(2):150–174.
- Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, 101(5):1160–1179.
- Luke Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorical Grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Shiqi Zhang and Peter Stone. 2015. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*.

Supplementary Material

A Improvement in parser accuracy

The following experiment is an attempt to quantify the accuracy of the parsers after training from dialog. This was done by hand-annotating the semantic forms for commands from the test set used for the first experiment. The results can be seen in table 7. The parsers are evaluated in terms of $Recall@1$, which is the fraction of times the correct parse is the top parse predicted by the parser, and $Recall@10$, which is the fraction of times the correct parse occurs in the top 10 parses predicted by the parser.

Learning involved	$Recall@1$	$Recall@10$
None	0.564	0.611
Only parser	0.588	0.671 *
Only dialog	0.564	0.623
Parser & dialog - <i>full</i>	0.588	0.647 ^
Parser & dialog - <i>batchwise</i>	0.576	0.670 *

Table 7: Comparison of performance of initial parser and parsers after updating various components, on paired commands and semantic forms. * indicates that the difference in performance between this and the *Initial* parser on the same metric is statistically significant according to a paired t-test with $p < 0.05$ and ^ indicates that the difference is trending significance ($p < 0.1$)

As expected, we observe that the initial parser (no learning) and the parser from the system performing only dialog learning, perform worse than the others, as the other systems update the parser used by these. The parser of the system performing only dialog learning is in fact a copy of the initial parser and was included only for completeness. Any difference in their performance is due to randomness. The parsers updated from dialogs improve in accuracy but the differences are found to be statistically significant only on $Recall@10$. The modest improvement is unsurprising given that the supervision provided is both noisy and weak. However, as seen in the main paper, even this modest improvement is sufficient to improve overall dialog success.

B Importance of multiple parse hypotheses

Many NLP systems typically return a list of top- n hypotheses, including semantic parsers. We use

the entire beam of top- n parses when updating the state. This is expected to be beneficial in cases where that the correct hypothesis is not the top ranked but present in this beam. The following experiment demonstrates that using multiple parses when updating the state improves overall dialog success. We compared an agent that used the same parser and policy as in the batchwise training but only the top ranked parse from the parser to update its state, as opposed to a beam of parses when updating its state. These two systems differed in no other components.

Number of parses considered	% successful dialogs	Dialog length
1	0.59	9.17
10	0.64	12.18

Table 8: Comparison of an agent using only the top hypothesis from the semantic parser and another using the top 10 parses. All differences are statistically significant according to an unpaired t-test with $p < 0.05$.

Table 8 shows the usefulness of considering multiple hypotheses from the semantic parser. As expected, the agent using multiple parses performs the correct action a significantly higher fraction of times. The system using a single hypothesis has a shorter average length among its successful dialogs because it rarely succeeds in more complicated dialogs where the system needs repeated clarification or answers to multiple specific questions.

Unsupervised AMR-Dependency Parse Alignment

Wei-Te Chen

Department of Computer Science
University of Colorado Boulder
weite.chen@colorado.edu

Martha Palmer

Department of Linguistics
University of Colorado Boulder
martha.palmer@colorado.edu

Abstract

In this paper, we introduce an Abstract Meaning Representation (AMR) to Dependency Parse aligner. Alignment is a preliminary step for AMR parsing, and our aligner improves current AMR parser performance. Our aligner involves several different features, including named entity tags and semantic role labels, and uses Expectation-Maximization training. Results show that our aligner reaches an 87.1% F-Score score with the experimental data, and enhances AMR parsing.

1 Introduction

Abstract Meaning Representation (AMR) (Banasescu et al., 2013) is a semantic representation that expresses the logical meaning of English sentences with rooted, directed, acyclic graphs. AMR associates semantic concepts with the nodes on a graph, while the relations are the label edges between concept nodes. Meanwhile, AMR relies heavily on predicate-argument relations from PropBank (Palmer et al., 2005), which share several edge labels. The representation also encodes rich information, like semantic roles (all the “ARGN” tags from PropBank), named entities (NE) (“person”, “location”, etc., concepts), wiki-links (“:wiki” tags), and co-reference (reuse of variables, e.g., p). An example AMR in PENMAN format (Matthiessen and Bateman, 1991) is shown in Figure 1.

The design of an AMR to English sentence aligner is the first step for implementation of an AMR parser, since AMR annotation does not contain links between each AMR concept and the original span of words. The basic alignment strategy is to link the AMR tokens (either concepts or edge labels) with their corresponding

```
(j / join-01
 :ARG0 (p / person :wiki -
 :name (p2 / name
 :op1 "Pierre" :op2 "Vinken")
 :age (t / temporal-quantity :quant 61
 :unit (y / year)))
 :ARG1 (b / board
 :ARG1-of (h / have-org-role-91
 :ARG0 p
 :ARG2 (d2 / director
 :mod (e / executive :polarity -))))
 :time (d / date-entity :month 11 :day 29))
```

Figure 1: The AMR annotation of sentence “Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.” in PENMAN format

span of words. Another strategy is to find the alignment from an AMR concept to a word node in a dependency parse tree, the goal of this paper. A dependency parse tree is a good structure for attaching more information, e.g. named entity tags, lemma, and semantic role labels, etc., and provides richer syntactic information than the span of words. An alignment between an AMR concept and a dependency node represents a correspondence between the meaning of this concept and its child concepts and the phrase governed by the dependency node (i.e., head word). An example alignment is shown in Figure 2. For example, the word node “Vinken” on the dependency parse side in Figure 2 links to the lexical concept of “Vinken” and, furthermore, links to the “ $p2/name$ ” and the “ $p/person$ ” concepts since “Vinken” is the head of the named entity “Pierre Vinken” and the head of the whole noun phrase “Pierre Vinken, 61 years old.”. In our work, we use Expectation-Maximization(EM) (Dempster et al., 1977) to train different feature probabilities, including rule-based features, lexical forms, relation labels, named entity tags, semantic role

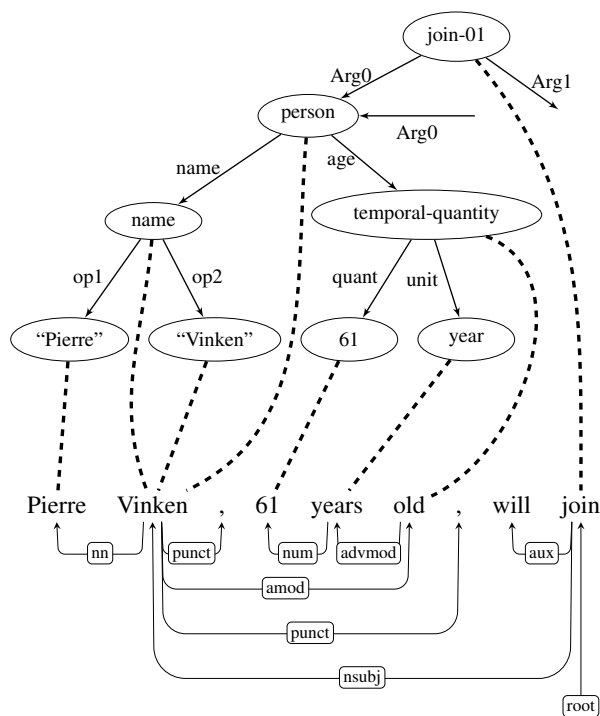


Figure 2: The alignment between a subgraph of an AMR (top) and a dependency parse (bottom) for the “Pierre Vinken” sentence. Dashed lines link dependency parse nodes and corresponding concepts.

labels, and global features, etc. Then EM processing incorporates all the individual probabilities and estimates the final alignments.

We will describe AMR-English sentence alignment in general, and review related work, in Section 2. Then the descriptions of our AMR-dependency parse features and alignment model are in Section 3. Our beam-search decoder is described in Section 4. Our experimental results are presented in Section 5, followed by our conclusion and discussion of future work (Section 6).

2 AMR-English Sentence Aligner

A preliminary step for an AMR parser is aligning AMR concepts and the original spans of words. JAMR (Flanigan et al., 2014) includes a heuristic alignment algorithm between AMR concepts and words or phrases from the original sentence. They use a set of alignment rules, like named entity, fuzzy named entity, data entity, etc., with a greedy strategy to match the alignments. This aligner achieves a 90% F_1 score on hand aligned AMR-sentence pairs. On the other hand, the ISI Aligner (Pourdamghani et al., 2014) presents a

generative model to align AMR graphs to sentence strings. They propose a string-to-string alignment model which transfers the AMR expression to a linearized string representation as the initial step. Their training method is based on the IBM word alignment model (Brown et al., 1993) but they modify the objective function of the alignment model. IBM Model-4 with a symmetric method reaches the highest F_1 score, 83.1%. When separating the alignments into roles (edge labels) and non-roles (concepts), F_1 scores are 49.3% and 89.8%, respectively. In Werling’s AMR parser (Werling et al., 2015), they conceive of the alignment task as a linear programming relaxation of a boolean problem. The objective function is to maximize the sum of action reliability. Each concept is constrained to align to exactly one token in a sentence. This ensures that only adjacent nodes or nodes that share the same title refer to the same token. They hand-annotate 100 AMR parses, and their aligner achieves an accuracy of 83.2%. By providing alternative alignments to their graph-based AMR parser, their aligner achieves a better Smatch score than JAMR’s aligner.

However, two transition-based parsers which parse dependency parse tree structures into AMRs, e.g., the CAMR system (Wang et al., 2015; Wang et al., 2016) and the RIGA system (Barzdins and Gosko, 2016), tie for the best results in SemEval-2016 task 8 (May, 2016). It is important to note that the JAMR aligner was not designed to align between a dependency word node and an AMR concept where its alignment F_1 score is only 69.8% (see Section 5.2). In order to deal with this problem, (Chen, 2015) proposed a preliminary aligner which estimates alignments by learning the feature probabilities of lexical (surface) forms, relations, named entities and semantic roles jointly. Besides the objective to obtain alignment between AMR concepts and original word spans, the estimation of these feature probabilities is also useful for further development of the AMR parser with these initial models. In our paper, we extend their previous work by adding rule-based and global features, and adding a beam-search algorithm at decoding time.

3 AMR-Dependency Parse Aligner

Our approach is an AMR-to-Dependency parse aligner, which represents one AMR as a list of

Concepts $C = \langle c_1, c_2, \dots, c_{|C|} \rangle$, and the corresponding dependency parse as a list of dependency word nodes $D = \langle d_1, d_2, \dots, d_{|D|} \rangle$. An alignment function a is designed to produce exactly one alignment to a dependency node d_{c_j} for each concept c_j , within a single sentence. Alternatively, we can view a as a mapping function that accepts one input variable concept c_j and outputs a dependency node d_{c_j} with which c_j is aligned. A is the alignment set that contains all different a_l that cover possible alignments within C and D . Our model adopts an asymmetric alignment direction, where one concept maps to exactly one dependency parse node, and each dependency parse node can be aligned by zero to multiple concepts. We denote dependency node d_c linked by concept c as $d_c = a(c)$. c^p is the parent concept of concept c , while $c^{s_1}, c^{s_2}, \dots, c^{s_k}$ are the k child concepts of concept c .

3.1 Features

3.1.1 Basic Features

Several of the AMR concepts use the word form directly. For example, the concept “join-01” in Figure 1 would align to the dependency node “join” naturally. Similarly, the leaf concepts usually align to identical terms in the dependency parse. In Figure 1, the names “Pierre” and “Vinken” are aligned to their word forms on the dependency parse leaves. Therefore, we design a straightforward rule-based probability, P_{rule} , which catches the appearance of the surface form. $P_{rule}(c, d_c)$ is defined as the probability that the matching type for a given concept c and dependency node d_c are linked. The different types of rules, e.g., word, lemma, numbers, and date, etc., and their proportional applicability to both AMR concepts and leaves are listed in Table 1. For example, the rule “Date” type aligns concept “11” with word node “November” in Figure 1, while “Numbers” aligns concept “5” with word node “five”. P_{rule} decides which match type to apply by following a greedy matching strategy.

3.1.2 External Features

To capture alignments for concepts which do not match any of the above basic rules, we design the following four external feature probabilities:

$$P_{Lemma}(c, d_c) = P(c|Word(d_c))$$

Lemma Probability represents the likelihood that a concept c aligns to a dependency word d_i . For

	Match Type	at Concept	at Leaf
(1)	Word	45.2%	73.4%
(2)	Word (case insensitive)	-	0.9%
(3)	Lemma (case insensitive)	10.8%	0.3%
(4)	Partial match with word	6.1%	8.2%
(5)	Partial match with lemma	0.2%	0.3%
(6)	Numbers	-	3.1%
(7)	Ordinal Numbers	-	2.8%
(8)	Date	-	4.3%
(9)	Others	37.7%	6.5%

Table 1: The rules and distribution of basic match types

example, in Figure 3a, the concept $c = \text{“temporal-quantity”}$ is highly likely to align to the word node $d_c = \text{“old”}$ since “old” is usually the head word of a phrase expressing age (“61 years old” here). Also, *have-org-role-91* can align to the word node “director” since “director” appears quite often with *have-org-role-91* (defined as roles in organizations). Besides, some special leaf concepts, like “:polarity -” (negative), and “:mode expressive” (which is used to mark exclamational word), also rely on this feature rather than the basic rules.

$P_{rel}(c, d_c, d_{c^p}) = P(AMRLabel(c)|Path(d_c, d_{c^p}))$
Relation Probability is the conditional probability of the AMR relation label of c , given the parse tree path between d_c and d_{c^p} , where d_c and d_{c^p} represent the dependency nodes that are aligned by c and c^p , respectively. Parse tree path is the concatenation of all dependency tree and direction labels through the tree path between d_c and d_{c^p} . For example, the relation probability of $c = \text{61}$, $d_c = \underline{61}$, and $d_{c^p} = \underline{old}$ in Figure 3b is $P(\text{quant}|\underline{advmod} \downarrow \text{num} \downarrow)$. A parse tree path is a useful feature for extracting relations between any two tree nodes, e.g., Semantic Role Labeling (SRL) (Gildea and Jurafsky, 2002) and relation extraction (Bunescu and Mooney, 2005; Kambhatla, 2004; Xu et al., 2015), so we add relation probability to our model.

$$P_{NE}(c, d_c) = P(c|NamedEntity(d_c))$$

Named Entity Probability is the probability of the concept c conditioned on different named entity types (e.g., PERSON, DATE, ORGANIZA-

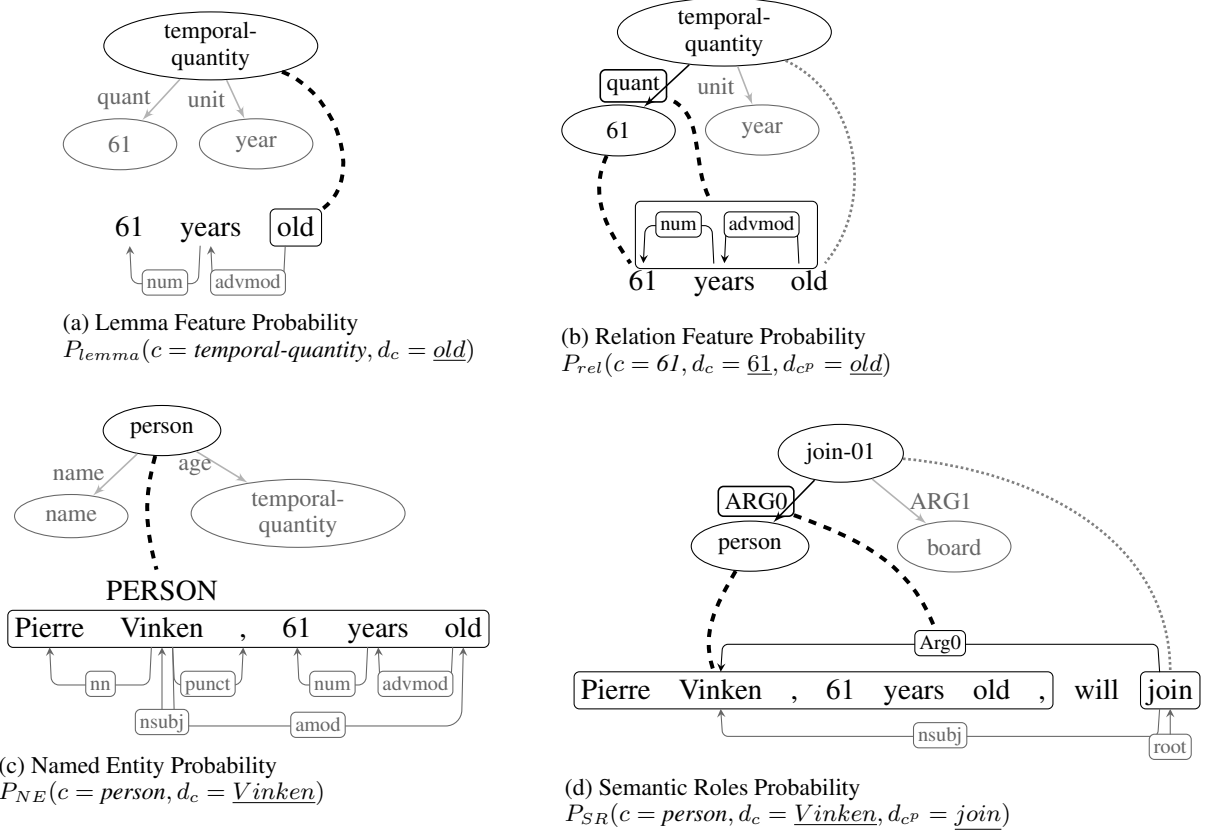


Figure 3: The sample of feature probabilities that are used in our aligner. Dashed lines link AMR concepts (top) and corresponding dependency parse nodes (bottom), while dense dashed lines link the parent AMR concepts and its corresponding dependency parse nodes.

TION, etc.). *NamedEntity*(d) indicates the named entity type of the phrase with d as the head word. For example, after named entity recognition (NER) tagging, the label assigned to “PERSON” is the dependency parse tree node “Vinken”. So the named entity probability of $P_{NE}(c = \text{person}, d_c = \text{Vinken})$ in Figure 3c is $P(\text{person} \mid \text{PERSON})$. Since AMR contains a large amount of named entity information, we assume that a feature based on an external named entity module should improve the alignment accuracy.

$$P_{SR}(c, d_c, d_{cp}) = P(\text{AMRLabel}(c) \mid \text{SemanticRole}(d_{cp}, d_c))$$

Semantic Role Probability is the conditional probability of the AMR relation label of c , given the semantic role d_c if d_{cp} is a predicate and d_c is d_{cp} ’s argument. If a predicate-argument structure does not exist between d_{cp} and d_c , the semantic role probability is omitted. For example, in Figure 3d, the semantic role probability of $P_{SR}(c = \text{person}, d_c = \text{Vinken}, d_{cp} = \text{join})$ is equal to $P(\text{ARG0} \mid \text{Arg0})$. Since AMR depends heavily on predicate-argument relations, external

predicate-argument information from an external SRL system should enhance the overall alignment accuracy.

The above four feature probabilities are learned by the EM algorithm (Section 3.2).

3.1.3 Global Feature

The above basic and external features capture local alignment information. However, to make sure that a concept is aligned to the correct phrase head word which represents the same sub-meaning, we need a global feature to calculate coverage. The design of our concept coverage feature is as follows:

$R_{CC}(c)$ **Overlapping Ratio** of the child concept aligned phrases to their parent concept aligned phrases plus the non-covered penalty. This ratio is defined as:

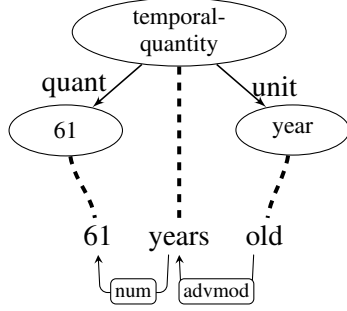


Figure 4: A sample of incorrect alignment. We use this sample to calculate its overlapping ratio (R_{cc}) let $c = \text{temporal-quantity}$

$$\begin{aligned}
 W(c) &= \{\underline{61}, \underline{year}\} \\
 W_{child}(c) &= \{\underline{61}\} \cup \{\underline{61}, \underline{year}, \underline{old}\} \\
 &= \{\underline{61}, \underline{year}, \underline{old}\} \\
 W_{child}(c) \cap W(c) &= \{\underline{61}, \underline{year}\} \\
 pen(c) &= \exp(-|\{\underline{old}\}|) = 0.37 \\
 R_{cc}(c) &= \left(\frac{2}{3}\right) \times pen(c) = 0.37
 \end{aligned}$$

$$\begin{aligned}
 R_{CC}(c) &= \frac{|W_{child}(c) \cap W(c)|}{|W(c)|} \times pen(c) \\
 W(c) &= d_c
 \end{aligned}$$

$$\begin{aligned}
 W_{child}(c) &= \bigcup_{c^s i \in child(c)} d_{c^s i} \\
 pen(c) &= \exp(-|W_{child}(c) \setminus (W_{child}(c) \cap W(c))|)
 \end{aligned}$$

where W refers to the set of words that the aligned dependency word node contains. The first term of R_{CC} ensures the child concepts contain the largest possible subspans of the parent concept span. The non-covered penalty term (pen) is to prevent a child concept from aligning to a word node that contains a larger word span than the child’s parent concept. The pen term will increase exponentially if child concepts align to a larger word span. The back slash term “ \setminus ” refers to set subtraction. We take Figure 4 as an example of an incorrect alignment example where the concept “*temporal-quantity*” aligns to “*year*” and the concept “*year*” aligns to “*old*”, the overlapping ratio of this alignment is 0.37 since it suffers a penalty. As we compare it with the correct alignment in Figure 3b, the overlapping ratio of this alignment is 0.67, which is much higher than the incorrect one.

3.2 Training with EM Algorithm

The objective function of our AMR-to-Dependency Parse aligner is listed as follows: Since our long term goal is to design a dependency parse to AMR parser, we define the objective function L_θ as the probability that dependency parses transfer to AMR graphs for the AMR-to-Dependency Parse aligner:

$$\theta = \operatorname{argmax} L_\theta(\text{AMR}|\text{DEP}) \quad (1)$$

$$\begin{aligned}
 L_\theta(\text{AMR}|\text{DEP}) &= \prod_{\substack{(C,D,A) \\ \in \mathbb{S}}} P(C|D) \\
 &= \prod_{\substack{(C,D,A) \\ \in \mathbb{S}}} \sum_{a \in A} P(C, a|D) \quad (2)
 \end{aligned}$$

$$P(C, a|D) = \prod_{j=1}^{|C|} P(c_j | d_{c_j} = a(c_j), d_{c_j^p} = a(c_j^p)) \quad (3)$$

where $\theta = (P_{lemma}, P_{rel}, P_{NE}, P_{SR})$ is the set of feature probabilities (parameters) we want to estimate, alignment set A is the latent variable we want to observe, and \mathbb{S} is the training sample that contains a set of tuples (C, D, A) , where C and D are a $\langle \text{AMR}, \text{dependency parse} \rangle$ pair and A is their alignment combination set. In equation (3), the probability that dependency tree D translates to AMR C with an alignment combination a is equal to the product of all probabilities that concept c_j in C aligns to dependency node d_{c_j} and c_j^p aligns to dependency node $d_{c_j^p}$.

3.2.1 Expectation-Step

The E-Step estimates all the different alignment probabilities of an input AMR and dependency parse pair by giving the product of feature probabilities. The alignment probability can be calculated using:

$$P(a|C, D) = \prod_{j=1}^{|C|} \frac{P(c_j | d_{c_j}, d_{c_j^p})}{\sum_{l=1}^{|D|} \sum_{i=1}^{|D|} P(c_j | d_i, d_l)} \quad (4)$$

$$\begin{aligned}
 P(c_j | d_i, d_l) &= P_{rule}(c_j, d_i) \times P_{lemma}(c_j, d_i) \\
 &\quad \times P_{rel}(c_j, d_i, d_l) \times P_{NE}(c_j, d_i) \times P_{SR}(c_j, d_i, d_l) \quad (5)
 \end{aligned}$$

The alignment probability is equal to the product of all tuple (c, d_c, d_{c^p}) ’s aligning probabilities.

P_{rule} is obtained by a simple calculation from the development set, while P_{lemma} , P_{rel} , P_{NE} , and P_{SR} are initialized uniformly before the first round of E-step. And these feature probabilities will be updated during the M-step.

3.2.2 Maximization-Step

In the M-Step, feature probabilities are re-estimated by collecting the count of all AMR-dependency parse pairs. The count of lemma (cnt_{lemma}), relation (cnt_{rel}), named entity (cnt_{NE}), and semantic role (cnt_{SR}) features are the normalized counts that are collected from the accumulating probability of all possible alignments from the E-step. Here we take the derivation of cnt_{lemma} as an example. cnt_{rel} , cnt_{NE} , and cnt_{SR} can be obtained with similar equations:

$$cnt_{lemma}(c|Word(d_c); C, D) = \sum_{a \in A} \frac{P(c|d_c, d_{c_p})}{\sum_{i=0}^{|D|} \sum_{l=0}^{|D|} P(c|d_i, d_l)}$$

After we collect all counts for different features, the four feature probabilities, P_{lemma} , P_{rel} , P_{NE} , and P_{SR} , are updated with their feature counts. Here we show the update of P_{lemma} as an example. The rest of feature probability updates can be derived in the same way:

$$P_{lemma}(c, d) \leftarrow \sum_{\substack{C \in AMR, \\ D \in DEP}} \frac{cnt_{lemma}(c|Word(d); C, D)}{\sum_{j=1}^{|C|} cnt_{lemma}(c_j|Word(d); C, D)}$$

After this, we apply the newer feature probabilities to recalculate alignment probabilities in the E-step again. EM iterates the E and M-steps until convergence or certain criteria are met.

4 Decoding

At decoding time, we want to find the most likely alignment a for the given $\langle C, D \rangle$. By applying Equations (4) and (5), we define the search for alignments as follows:

$$\operatorname{argmax}_a P(a|C, D) = \operatorname{argmax}_a \prod_{j=1}^{|C|} R_{CC}(c_j) * P(c_j|d_i = a(c_j), d_l = a(c_j^p))$$

This decoding problem finds the alignment a that maximizes the likelihood, which we define in

		Sent.	Token	# of NE	# of Arg.
Gold	train	8,276	176,422	3,750	58,520
	dev.	409	8,695	415	2,574
	test	415	8,786	401	3,107
All	train	39,260	649,219	43,715	260,979
	dev.	409	8,695	580	2,574
	test	415	8,786	401	3,107

Table 2: The data split of the LDC DEFT AMR corpus. *Gold* refers to the sentences also appearing in OntoNotes 5.0 with gold annotations, while *All* refers to all sentences in DEFT AMR corpus with dependency parses, named entities, and semantic roles generated by ClearNLP. Number of tokens, named entities, and arguments(Arg.) in each data set are also presented

Equation (5). The overlapping ratio (R_{CC}) is introduced to the likelihood function to ensure that a parent concept covers a wider word span range than its child concepts. A beam search algorithm is designed to extract the target alignment without exhaustively searching all of the candidate alignments (which has a complexity of $O(|D|^{|C|})$.) The beam search starts from leaf concepts and then walks through parent concepts after their child concepts have been traversed. When we go through concept c_j , we need to consider all the following likelihoods: 1) the accumulated likelihood for aligning to any dependency word node d_{c_j} from all the child concepts of c_j , and 2) the product of P_{lemma} , P_{NE} , P_{rel} , P_{SR} , and R_{CC} for c_j . Instead of using during training, R_{CC} is only applied during decoding time. The probabilities are obtained simply from the product of all the above likelihoods. We keep the top- $|b|$ alignment probabilities and their aligned dependency node d_{c_j} for each c_j until we reach the root concept, where $|b|$ is the beam size. Finally we can trace back and find the most likely alignment.

The running time for the beam search algorithm is $O(|b| * |C| * |D|^2)$.

5 Experiments and Results

5.1 Experimental Data

The LDC DEFT Phase 2 AMR Annotation Release 2.0¹ consists of AMRs with English sentence

¹LDC DEFT Phase 2 AMR Annotation Release 2.0, Release date: March 10th, 2016. <https://catalog.ldc.upenn.edu/LDC2016E25>

pairs. Annotated selections from various genres (including newswire, discussion forum, other web logs, and television transcripts) are available, for a total of 39,260 sentences. This release uses the PropBank Unification frame files (Bonial et al., 2014; Bonial et al., 2016). To generate automatic dependency parses for all DEFT AMR Release data, we use ClearNLP (Choi and Mccallum, 2013) to produce dependency parses. ClearNLP also labels semantic roles and named entity tags automatically on the generated dependency parses. This data set is named “All”. To compare the effect of applying automatic dependency parses to our aligner with gold dependency parses, we select the sentences which appear in the OntoNotes 5.0² release as well. The OntoNotes data contains TreeBanking, PropBanking, and Named Entity annotations. OntoNotes 5.0 also uses PropBank Unification frame files for PropBanking. This data set, containing a total of 8,276 of selected AMRs and their dependency parses from OntoNotes, is named “Gold”. To generate the development and test set, we manually align the AMR concepts and dependency word nodes. Since the manual alignment is time-consuming, “Gold” and “All” data share the same development/test set. Table 2 presents the statistics for the experimental data.³

5.2 Experiment Results

We run EM for 50 iterations and ensure the EM model converges. Afterwards, we use our decoding algorithm to find the alignments that maximize the likelihood. The test set data is used to evaluate performance.

We first evaluate the performance of our system with the external features added incrementally. Table 3 indicates the results. By running with the “Gold” data, the only feature that improves significantly over the baseline (rule-based and lexicon features only) is the semantic role feature. The named entity feature actually hurts performance. On the other hand, all the features contribute to the F-Score incrementally for “All”. Again, the semantic role feature still has the most positive impact against other features, and a significant improvement over the baseline.

As we compare the F_1 score on training with

²LDC OntoNotes Release 5.0, Release date: October 16th, 2013 <https://catalog.ldc.upenn.edu/LDC2013T19>

³The manually aligned data and our aligner will be available after this paper gets accepted

Data	Feature	P	R	F-Score
Gold	L	84.0	85.0	84.5
	L + S	85.2	86.3	85.7
	L + S + R	82.8	83.8	83.3
	L + S + R + N	80.9	81.9	81.4
All	L	84.9	85.4	85.1
	L + S	85.7	87.4	86.5
	L + S + R	85.8	87.7	86.7
	L + S + R + N	86.3	88.0	87.1

Table 3: Incremental Feature Contributions for different features: L : lemma; R : relation; N : NE; S : semantic role.

“All” and “Gold” data set, training with “All” outperforms training with “Gold” data in all different feature combinations. We believe there are two reasons for this. First, the “All” data contains richer information than the “Gold” data. “All” has double the sentence size of “Gold”, and proportionally more named entity labels. Second, the automatic dependency parses do not hurt the performance of our aligner very much. We believe that our unsupervised alignment model works better with more data, even without access to gold standard dependency parses.

We then compare our aligner with three other aligners: JAMR, another version of unsupervised alignment (Chen, 2015), and ISI. To make them fit our test data, we design a heuristic method to force every unaligned concept (e.g., named entity and “:polarity -” concepts) to align to a dependency word node according to rule-based and global features (see Section 3.1). The alignment is counted as a correct match when the concept aligns to either the head word or the partial word span of a phrase. The alignments from concept relation to word span (apply in ISI) are discarded in our task. The results of the experiment are shown in Table 4. Our aligner achieves the best F_1 score in both the “All” and “Gold” data sets, as it should, since it is designed to align AMRs to dependency parses, as was the Chen aligner. Our aligner performs better than the Chen aligner by around 28% in F_1 score. We can conclude that the addition of rule-based feature, global features, and beam-search in decoding time helps the alignment task substantially.

5.3 Apply to AMR Parsing

To evaluate how alignment can enhance AMR parsing, we compare the parsing performance of

Data	Aligner	P	R	F-Score
Gold	Chen 2015	61.1	53.4	57.0
	JAMR	78.5	62.8	69.8
	ISI	78.6	71.4	74.9
	Ours	85.2	86.3	85.7
All	Chen 2015	62.4	55.5	58.7
	JAMR	80.2	65.9	72.4
	ISI	80.4	74.9	77.6
	Ours	86.3	88.0	87.1

Table 4: Results of different alignment models

the CAMR parser with different alignments produced by JAMR, ISI, and our aligner. To make the alignments fit the CAMR parser, we convert both ISI and our alignments to the original JAMR alignment format, word span to AMR concept. We get rid of the “:wiki” tag, which links the named entity to its Wikipedia page, to simplify the parsing task since we think the Wikify task (Mihalcea and Csomai, 2007) is basically different from the AMR parsing task. Smatch v2.0.2 is used to evaluate AMR parsing performance (Cai and Knight, 2012). The evaluation script is obtained from the SemEval 2016 Task 8 website⁴.

A comparison of parsing results is given in Table 5. We first train the parser with “Gold” Standard dependency parses and alignments from the different aligners. Results show that our aligner improves by a 2% F_1 score over the two other aligners. Then we train the AMR Parser system with the “All” data set. The dependency parses attached with semantic roles and named entities generated by ClearNLP are also provided to CAMR as training data. CAMR use dependency parsing results from Stanford dependency parser (Klein and Manning, 2003) by default. Our aligner still achieves slightly better performance than the other two. Modifying the AMR parser to take advantage of parse node-concept alignments could potentially result in greater improvement, since CAMR takes the input alignments as word span to AMR concept.

5.4 Error Analysis

To further understand the advantages and the disadvantages of our model, we go through all incorrect alignments and manually categorize 40% of them into different error types, with their propor-

⁴<http://alt.qcri.org/semeval2016/task8/index.php?id=data-and-tools>

Data	Aligner	P	R	F-Score
Gold	JAMR	62.2	61.0	61.1
	ISI	65.3	63.9	64.5
	Ours	68.6	64.2	66.4
All	JAMR	64.2	63.0	63.1
	ISI	66.1	65.1	65.6
	Ours	68.1	64.7	66.7

Table 5: Comparison of using different alignments with CAMR Parser.

tion:

Automatic Parsing Errors - 3.8%: ClearNLP has a 92.96% unlabeled attachment score on the Penn English Treebank evaluation set (Marcus et al., 1993), Section 23, for dependency parsing. Therefore, when training our aligner on the “All” data set with dependency parses, named entities, and semantic roles generated by ClearNLP, incorrect parses occasionally show up. Since NE and semantic roles are attached to dependency parses, incorrect dependency parses cause additional NE and semantic roles alignment errors, on top of the dependency parse alignment errors.

Long Distance Dependencies - 14.2%: Long sentences with long distance dependencies always bring difficulty to NLP parsing tasks. Experimental results show that our model runs into troubles when nearby concepts align to dependency nodes which are far from each other. Co-reference is an example that is highly likely to align to long distance dependencies, and our model can not deal with it well.

Duplicate Words - 17.4%: When two identical concepts align to different word nodes, our model is confused by duplicate words. In Figure 5, there are two “first”s in the sentence. One refers to “first 6 rounds”, and the other refers to “first position”. However, our model faultily aligns both *ordinal-entity* concepts to the same “first” word node. Our model did not distinguish these two ordinal-entities since the lexicon and named entity tags of the two “first”s are identical.

Meaning Coverage Errors - 40.4%: We define a good alignment as a concept that aligns to the correct phrase head word which represents the same sub-meaning. So instead of aligning to a concept’s word lexicon, sometimes a concept aligns to its parent node (head word). However, the lexicon features dominate the alignment probability in our

```

(a / and
:op1 (o / occupy-01
:ARG0 (p3 / person
:name (n / name
:op1 "Mingxia" :op2 "Fu"))
:ARG1 (p4 / position
:ord (o3 / ordinal-entity :value 1))
:op2 (o2 / occupy-01
:ARG0 (p / person
:name (n2 / name
:op1 "Bin" :op2 "Chi"))
:ARG1 (p2 / position
:ord (o4 / ordinal-entity :value 3))
:mod (r / respective)
:time (r3 / round-05 :quant 6
:ARG1 (c / compete-01)
:ord (o5 / ordinal-entity :value 1)))

```

Figure 5: The AMR annotation of sentence “In the first 6 rounds of competition, Mingxia Fu and Bin Chi are occupying the first and third positions respectively”

E-M calculation. That causes our model to tend to align a concept with its word form instead of its head word. For example, English light verb constructions (LVCs), e.g., *take a bath*, are thought to consist of a semantically general verb and a noun that denotes an event or state. AMR representation always drops light verb and uses eventive noun as concept. Our model sometimes aligns this eventive noun concept to its nominal word node, which is incorrect since the light verb on dependency parse covers the same sub-meaning and should be aligned.

6 Conclusion and Future Work

In this paper, we present an AMR-Dependency Parse aligner, which estimates the feature probabilities by running the EM algorithm. It can be used directly by AMR parser. Results show that our aligner performs better than other aligners, and improves AMR parser performance. The latent probabilities that we obtain during training, i.e., all the external feature sets, could also potentially benefit a parser. We plan to develop our own AMR parser, which will apply these external feature sets as the basic model. We also plan to continue to perfect our aligner via tuning the feature weights and learning techniques, and adding new features, like word embeddings and WordNet features.

Acknowledgments

We gratefully acknowledge the support of the National Science Foundation Grants 0910992 IIS:RI: Richer Representations for Machine Translation, and NSF IIA-0530118 PIRE (a subcontract from Johns Hopkins) for the 2014 Frederick Jelinek Memorial Workshop for Meaning Representations in Language and Speech Processing, and funding under 2016 Summer Graduate School Fellowship of University of Colorado at Boulder. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.
- Guntis Barzdins and Didzis Gosko. 2016. RIGA at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. *CoRR*, abs/1604.01278.
- Claire Bonial, Julia Bonn, Kathryn Conger, Jena D. Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).
- Claire Bonial, Kathryn Conger, Jena D. Hwang, Aous Mansouri, Yahya Aseri, Julia Bonn, Timothy O'Gorman, and Martha Palmer. 2016. Current directions in english and arabic propbank. In Nancy Ide and James Pustejovsky, editors, *The Handbook of Linguistic Annotation*. Springer, Berlin.
- Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.
- Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05*, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics.

- Shu Cai and Kevin Knight. 2012. Smatch: an evaluation metric for semantic feature structures. submitted.
- Wei-Te Chen. 2015. Learning to map dependency parses to abstract meaning representations. In *Proceedings of the ACL-IJCNLP 2015 Student Research Workshop*, pages 41–46, Beijing, China, July. Association for Computational Linguistics.
- Jinho D. Choi and Andrew Mccallum. 2013. Transition-based dependency parsing with selectional branching. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.
- J. Flanigan, S. Thomson, J. Carbonell, C. Dyer, and N. A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*, Baltimore, Maryland, June. Association for Computational Linguistics.
- Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, September.
- Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June.
- C. Matthiessen and J.A. Bateman. 1991. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Communication in artificial intelligence. Pinter.
- Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1063–1073, San Diego, California, June. Association for Computational Linguistics.
- Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.
- Martha Palmer, Dan Guildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105, March.
- Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429. Association for Computational Linguistics.
- Chuan Wang, Xue Nianwen, and Pradhan Sameer. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.
- Chuan Wang, Nianwen Xue, Sameer Pradhan Pradhan, Xiaoman Pan, and Heng Ji. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*. Association for Computational Linguistics, June.
- Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *CoRR*, abs/1506.03139.
- Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Improving Chinese Semantic Role Labeling using High-quality Surface and Deep Case Frames

Gongye Jin*, Daisuke Kawahara, Sadao Kurohashi

Graduate School of Informatics, Kyoto University

Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan

jin@nlp.ist.i.kyoto-u.ac.jp, {dk, kuro}@i.kyoto-u.ac.jp

Abstract

This paper presents a method for improving semantic role labeling (SRL) using a large amount of automatically acquired knowledge. We acquire two varieties of knowledge, which we call surface case frames and deep case frames. Although the surface case frames are compiled from syntactic parses and can be used as rich syntactic knowledge, they have limited capability for resolving semantic ambiguity. To compensate the deficiency of the surface case frames, we compile deep case frames from automatic semantic roles. We also consider quality management for both types of knowledge in order to get rid of the noise brought from the automatic analyses. The experimental results show that Chinese SRL can be improved using automatically acquired knowledge and the quality management shows a positive effect on this task.

1 Introduction

Semantic role labeling (SRL) is regarded as a task that is intermediate between syntactic analysis and semantic analysis in natural language processing (NLP). The main goal of SRL is to extract a proposition from a sentence about *who* does *what* to *whom*, *when*, *where* and *why*. By using semantic roles, the complex expression of a sentence is then interpreted as an *event* and its *participants* (i.e., a predicate and arguments such as *agent*, *patient*, *locative*, *temporal* and *manner*). Unlike syntactic level surface cases (i.e., dependency labels such as subject and object), semantic roles can be regarded

as a deep case representation for predicates. Because of its ability to abstract the meaning of a sentence, SRL has been applied to many NLP applications, including information extraction (Christensen et al., 2010), question answering (Pizzato and Mollá, 2008) and machine translation (Liu and Gildea, 2010).

Semantically annotated corpora, such as FrameNet (Fillmore et al., 2001) and PropBank (Kingsbury and Palmer, 2002), make this type of automatic semantic structure analysis feasible by using supervised machine learning methods. However, supervised SRL methods have the following two major issues. Firstly, as a common issue in almost all the supervised approaches, it is expensive to enlarge manually annotated corpora to learn a more accurate model. Secondly, experiments show that automatic SRL systems strongly depend on syntactic information. In practice, these SRL systems suffer from errors propagated from the lower-level syntactic analyses, such as word segmentation, POS tagging, and dependency parsing. Although some studies use automatic analyses of unlabeled corpora to enrich the training data to solve the first problem (Fürstenaу and Lapata, 2009), accumulated errors in such automatic analysis inevitably cause negative effects. Especially, for some hard-to-analyze languages, such as Chinese, which is still difficult to precisely analyze word segmentations, the performance of SRL is always limited due to the above two problems.

In this paper, we focus on Chinese SRL and address the problems mentioned above by using high-quality knowledge automatically acquired from a large-scale raw corpus. We utilize two types of additional knowledge. The first type is compiled using automatic syntactic analysis (specifically, dependency parsing) and is named **surface case frames** which are not expressive in

*The first author is now affiliated with Canon IT Solutions Inc.

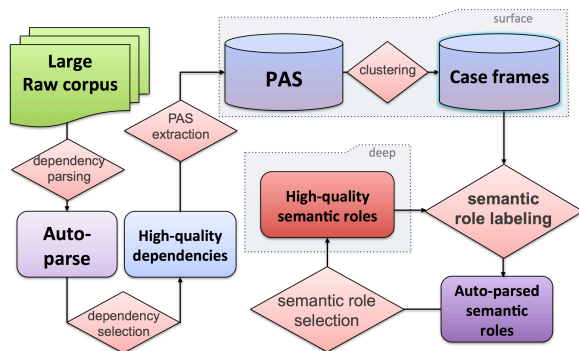


Figure 1: Overview of the framework

semantic level. In order to compensate the drawback of surface case frames, we also compile another type of knowledge using automatic semantic roles. We call this type of knowledge **deep case frames**. We illustrate the whole framework in Figure 1. The additional knowledge can provide not only syntactic information but also semantic information, both of which play crucial roles in SRL. Considering the inevitable noises from automatic analyses, we utilize an automatic selection method to select dependencies and semantic roles of high quality. In order to show that automatically extracted knowledge is beneficial and the quality management is indispensable, we compile both types of knowledge in different quality in our experiments and apply them to Chinese SRL.

2 Related work

The CoNLL-2009 shared task (Hajič et al., 2009) features a substantial number of studies on SRL that used Propbank as one of the resources. The participating systems can be categorized into two types: joint learning of syntactic parsing and SRL (Tang et al., 2009; Morante et al., 2009), which learns a unique model for syntactic parsing and SRL jointly. This type of framework has the ability to use SRL information in syntactic parsing for improvement, but needs a much larger search space for decoding. The other type is called SRL-only task (Zhao et al., 2009; Björkelund et al., 2009), which uses automatic morphological and syntactic information as the input in order to judge which token plays what kind of semantic role. Our work focuses on the second category of SRL. Our framework is based on those used by Björkelund et al. (2009) and Yang and Zong (2014).

There were several studies using additional knowledge to improve syntactic and semantic tasks. McClosky et al. (2006) used an addi-

tional unlabeled corpus to reduce data sparsity. In syntactic level of NLP, rich knowledge, such as predicate-argument structures and case frames, is strong backups for various kinds of tasks. Case frames, which clarify relations between a predicate and its arguments, can support tasks ranging from fundamental analysis, such as dependency parsing and word similarity calculation, to multilingual applications, such as machine translation. Japanese case frames have been successfully compiled (Kawahara and Kurohashi, 2006), where each case slot is represented as its case marker in Japanese such as ‘ga’, ‘wo’, and ‘ni’. For the case frames of other languages such as English and Chinese, because there are no such case markers that can help clarify syntactic structures, instead of using case markers like in Japanese, syntactic surface cases (i.e., subject, object, prepositional phrase, etc.) are used for argument representation (Jin et al., 2014). Case frames can be automatically acquired using a different method such as Chinese Restaurant Process (CRP) (Kawahara et al., 2014) for different languages. In our work, we employ such syntactic level knowledge, which uses surface cases as argument representation, to help SRL.

One basic idea of semi-supervised SRL is to automatically annotate unlabeled data using a simple classifier trained on original training data (Fürstenaу and Lapata, 2009). Since there is a substantial amount of error propagation in the SRL pipeline, the additional automatic semantic roles are not guaranteed to be of good quality. Also, some studies assume that sentences that are syntactically and lexically similar are likely to share the same frame-semantic structure (Fürstenaу and Lapata, 2009). This allows them to project semantic role information to unlabeled sentences using alignments. However, the computation of these alignments requires additional information such as word similarity, whose quality is language dependent. Less sparse features capturing lexical information of words can be also used for semi-supervised learning of SRL. Such lexical representation can be learned from unlabeled data (Bengio et al., 2003). Deschacht and Moens (2009) used word similarity learned from unlabeled data as additional features for SRL. Word embeddings have also been used in several NLP tasks including SRL (Collobert et al., 2011). Instead of using word-level lexical knowledge, our work uses syn-

tactic and semantic knowledge, i.e., case frames. Word embeddings can also be incorporated into our method but we leave this to our future work. Zapirain et al. (2009) used selectional preferences to improve SRL. This study is similar to our approaches but the quality of selectional preferences was not concerned at all.

3 SRL task description

In previous studies, SRL pipeline¹ can be divided into three main steps: predicate disambiguation (PD), argument identification (AI), and argument classification (AC). In the PD step, the main goal is to identify the “sense id” of each given predicate. The AI step mainly focuses on judging whether each argument is semantically related to each predicate in a sentence. Based on the results of the AI step, the AC step assigns a semantic role to each semantically related argument. Basically, the PD step and the AC step are regarded as multi-class classification problems while the AI step is a binary classification problem.

In the PD step, because the sense id for a certain predicate is meaningless for other predicates, classifiers for PD are trained separately for each predicate. We basically use the feature set proposed by Björkelund et al. (2009). During the prediction, there are some predicates which have not been seen in the training data. We label the sense of those unseen predicates using the default sense, which is ‘01’ in our work.

4 Applying high-quality surface case frames to SRL

4.1 High-quality dependency selection

Dependency parsing has been widely employed for knowledge acquisition related to predicate-argument structures. The dependency parsing performance determines the quality of acquired knowledge, regardless of target languages. Reducing dependency parsing errors and selecting high-quality dependencies are of primary importance. Jin et al. (2013) used a single set of dependency labeled corpus (a treebank), a part of which was used to train a base dependency parser. Another part of the labeled corpus was used to apply automatic dependency parsing. By comparing the

¹Predicate identification (PI) was not concerned in the experiments because we use the data from CoNLL-2009 shared task, in which the target predicates are given.

gold standard data and the automatic parses, correct dependencies were collected as positive examples and incorrect dependencies were collected as negative examples. Then selecting high-quality dependencies was regarded as a binary classification problem. To conduct such binary classification, they employed a set of basic features from Yu et al. (2008). In addition to these basic features, Jin et al. (2013) considered context features that are thought to affect parsing performance. Since the input for high-quality dependency selection method is a dependency tree, tree features are used to identify dependency quality. Also, some dependency parsers output the score of each dependency (i.e., edge confidence value) during the parsing process. They used the real value of the score as an additional feature. We first apply this approach to select high-quality dependencies from automatic parses.

4.2 High-quality surface case frame construction

After applying dependency parsing on a large-scale raw corpus, predicate-argument structures (PASs) are extracted using the high-quality dependencies. Arguments are represented by their dependency labels (i.e., subject, object, etc.) For each predicate, all the PASs are clustered into different case frames to reflect different semantic usages. We show an example of case frames for the verb ‘谢’ in Table 1, which has multiple meanings. ‘谢(1)’ is the case frame used to represent the sense of ‘withering of flower’. Similarly, the sense of ‘谢’ which means ‘to thank’ is represented by case frame ‘谢(2)’. ‘谢(3)’ is the case frame for the sense of ‘curtain call’. In other words, case frames are knowledge that solves word sense disambiguation (WSD) by clustering the PASs. We applied the CRP method described by (Kawahara et al., 2014) for clustering the high-quality PASs to compile high-quality case frames.

4.3 Surface case features for SRL

From the surface case PASs, we extract four types of additional features, for both AI and AC step. These features are described in the upper part of Table 2. We use binned values (i.e., high, middle and low) for all of the feature values calculated from the knowledge. More specifically, for each type of feature, we define the first, second and third tertile of all the feature values as low, middle and high correspondingly. Surface case

verb	surface case	instance with frequency in original corpus
谢(1)	nsubj	花儿(flower):14, 花(flower):22
	ad	都(all):16, 也(also):6
谢(2)	nsubj	你们(you):1
	dobj	您(you):8, 我(me):6
	ad	怎么(how):8, 多(very):1
谢(3)	nsubj	大战(battle):1
	dobj	幕(curtain):6
	ad	圆满(successfully):2, 也(also):1, 正式(officially):1
...		

Table 1: Examples of Chinese surface case frames

frames are clustered PASs according to each predicate’s semantic usage. Therefore, instead of utilizing all the predicate-argument structures, it is intuitive to use the predicate-argument structures only from the corresponding case frames. So we also create four types of features extracted from case frames. These features are listed in the lower part of Table 2.

Note that a case frame ID and a PropBank sense ID do not correspond to each other. In practice, the number of case frames is always larger than the number of sense in PropBank for each verb. As a result, a mapping process that aligns case frame id(s) to PropBank verb sense is applied. First, we assign automatic dependency labels to the PropBank corpus using the Stanford parser. We then calculate the similarity between a PropBank sense and a case frame by measuring the PAS similarity. As shown in the left part of Figure 2, for a certain predicate with a sense ID in PropBank, we represent the predicate in each sense by using the collection of all the instances in each syntactic role slot. Each predicate with a sense ID is then transformed into a vector space, which we name PAS vector. The same transformation is applied to case frames. Then the cosine similarity between vectors transformed from a PropBank sense and case frames is calculated. A PAS vector is the concatenation of each syntactic role vector. To form a syntactic role vector, we simply take the average of weighted summation of the word vectors within the case slot. Word vectors are acquired using word2vec² from the same raw corpus that we use for knowledge acquisition (see Section 7.1). In our experiments, we only used syntactic role “subj” (subject) and “dobj” (direct object) because

these two syntactic roles are considered to be relatively more informative.

5 Main problem of surface case frames

In previous work (Kawahara and Kurohashi, 2006), case frames for Japanese are composed of all the instances and their corresponding case marker. For example, all the instances in “ga” case are basically the “subject” of the given predicate. Instances in “wo” case are basically the “direct object” of the given predicate. Other cases like “ni” can indicate “location”, “time” or “direction”. During the automatic PAS extraction for Japanese, there are also ambiguous case makers that can represent multiple cases. The most common one, for example, is “wa” case in Japanese. This case marker always functions as a topic marker. The argument in “wa” case is normally emphasized as the topic of the sentence. It can be equal to either “ga” case or “wo” case, and sometimes “ni” case. To avoid such ambiguous cases, one can simply discard all the instances in “wa” case to make case frames more precise.

For languages that lack such case markers (e.g., English and Chinese), case frames are composed of automatic syntactic roles (Jin et al., 2014). Such syntactic roles include “subject”, “direct object”, “indirect object” and “prepositional phrases”. Such surface cases have limitations on case representation especially for Chinese. Take the following sentences as examples.

- (1) 苹果 (apples) / 我 (I) / 吃了 (eaten) / 很多 (a lot).
- (2) 我 (I) / 苹果 (apples) / 吃了 (eaten) / 很多 (a lot).
- (3) 我 (I) / 吃了 (eaten) / 很多 (a lot) / 苹果 (apples).
- (4) 我 (I) / 吃了 (eaten) / 很多 (a lot).

²<https://code.google.com/archive/p/word2vec/source/default/source>

feature	description
Freq	the co-occurrence frequency of a predicate-argument pair without considering the syntactic role of the argument
Pmi	the point-wise mutual information (PMI) value for each predicate-argument pair without considering the syntactic role of the argument
PAfreq	the frequency of a argument being a certain syntactic role of a predicate
PApmi	the PMI value of an argument with its syntactic role and the predicate
CFFreq	Freq value calculated only from within the corresponding case frames
CFPmi	Pmi value calculated only from within the corresponding case frames
CFPAfreq	PAFreq value calculated only from within the corresponding case frames
CFPami	PApmi value calculated only from within the corresponding case frames

Table 2: Surface case features for SRL

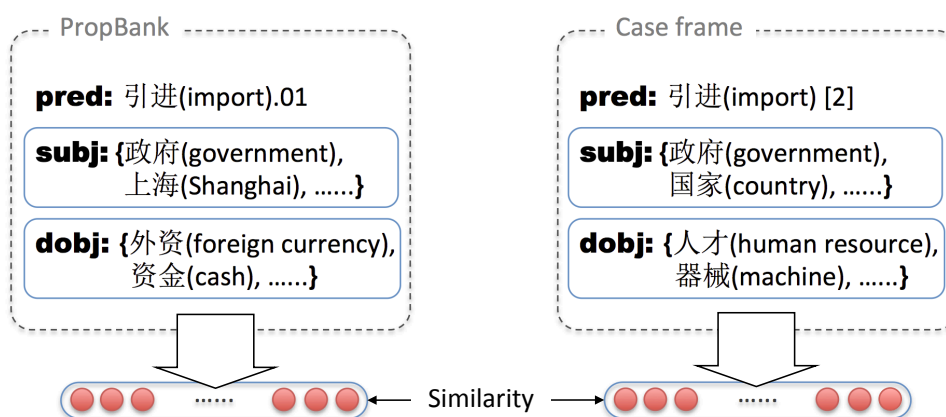


Figure 2: Overview of mapping case frames to PropBank sense

(5) 苹果 (apples) / 吃了 (eaten) / 很多 (a lot).

The first three sentences have the same meaning: “I have eaten a lot of apples.” However, as we can see from the sentences, the word “苹果 (apple)” which is a direct object of “吃了 (eaten)”, and the word “我 (I)” can be filled in various word orders. Also, because omissions occur frequently in Chinese, sentence 4 and 5 are also commonly used, which mean “I have eaten a lot” and “(I) have eaten a lot of apples”, respectively. Without considering the actual meaning of “我 (I)” and “苹果 (apples)”, both of them in sentence 4 and 5 are labeled as “subject” in the surface case representation following the syntactic grammar. If one tries to figure out which “subject” is actually in *Nominative Case* (which stands for the person/thing who provides the action) and which “subject” is in *Accusative Case* (which stands for the thing/person who receives/suffers from the action), it is always problematic because of the flexible word order and omission.

Although some studies found that applying simple mapping rules for *Nominative Case* and *Accusative Case* can achieve an overall high baseline for English, we found that this simple mapping cannot work well for Chinese. Here is an example which Chinese people are using for self-deprecating.

- (6) 中国 (Chinese) / 乒乓球 (table tennis) / 谁 (who) / 也 (ever) / 赢 (win) / 不了 (not): Nobody can win Chinese table tennis.
- (7) 中国 (Chinese) / 足球 (soccer) / 谁 (who) / 也 (ever) / 赢 (win) / 不了 (not): Chinese soccer cannot win anybody.

“Table tennis” and “soccer” should be labeled as *Accusative Case* and *Nominative Case* differently even though the predicate and the syntactic structure for both the sentences are identical.

Similar phenomena also occur in Japanese and make it difficult to analyze as well. However, in case of Japanese, it is possible to make use of the

morphemes attached to the predicate. For example, the following sentences are the Japanese translations for sentence 4 and 5.

- (8) 私が (I) / たくさん (a lot) / 食べた (eaten).
(9) りんごが (apples) / たくさん (a lot) / 食べられた (eaten).

There is always an additional morpheme (e.g., “られた”) attached to the predicate in order to indicate its voice. In the above example, sentence 8 can be regarded as active voice and sentence 9 is in passive voice. Unfortunately, Chinese is a language that lacks morpheme information. There are very few such markers that indicate the transitivity, voice and tense. This makes it almost impossible for a system to automatically recognize the ambiguous syntactic roles. To solve this problem, based on the syntactic analysis, we apply an SRL process to discover a deeper level case representation.

6 Applying high-quality deep case frames to SRL

6.1 High-quality semantic role selection

Similar to the previous work described in Jin et al. (2013), instead of using all the SRL outputs, we propose to use only automatically generated semantic roles of high quality.

In particular, the standard training section of the human-annotated data is used to train a base SRL model (which include three sub-models for predicate sense disambiguation (PD), argument identification (AI) and argument classification (AC)). Then, another part of the human-annotated data is used to apply SRL using the base model. From these results, we acquire training data for semantic role selection by collecting each semantic role. We then judge the correctness of each semantic role according to the gold standard annotations. All correct semantic roles are used as positive examples and the incorrect ones are used as negative examples for semantic role selection. Judging whether an automatic semantic role is reliable can be regarded as a binary classification problem. We use SVMs to solve this problem. We use the feature set for SRL described in Jin et al. (2015) as basic features. It contains predicate features that are extracted from the target predicate; argument features which are extracted from each candidate argument. We also use surface case frames, which have a positive effect on SRL, as additional knowledge.

6.2 High-quality deep case frame construction

Due to the major issues described in Section 5, case frames constructed using surface cases may be problematic. For example, for the predicate “吃 (eat)”, both the argument “苹果 (apple)” and “我 (I)” are assigned to the same surface case “subject”. If one tries to use this kind of surface case knowledge for tasks that require semantic information, such as machine translation (MT), it may lead to a performance drop. So we propose to construct deep case frames that are relatively more representative than the surface case frames. By the deep case, we mean using the semantic roles for case frame construction.

Compared to syntactic analysis, SRL is mainly used to clarify deeper-level semantic relations (e.g., [*who*] do [*what kind of*] thing to [*whom*] in [*what time*]) in the sentence, which has a better representation for predicate-argument relations. On the other hand, this task is always based on the tasks in preceding levels, such as morphological analysis and syntactic parsing. Especially, the information provided by syntactic parsing is crucial to achieve a good performance in SRL. An SRL system also suffers from the training data size issue as most of the machine learning approaches do. Extensive human efforts are required in order to construct such training data. Sometimes, the requirements for annotators can be higher than those for syntactic analysis. These factors along with the automatic analysis errors propagated from the lower-level analyses make it almost impossible for an SRL system to achieve a high performance.

For predicate identification (PI), we regard every word with a POS tag beginning with “V” as a predicate. The PD step in the SRL pipeline assigns a sense ID (frame ID) to each predicate. This is equivalent to the unsupervised clustering for surface case frames and thus no additional clustering process is required. After argument identification and argument classification, we only use these semantic roles with high reliability. For each predicate with different frame IDs, we collect all the high-quality semantic roles to compose the deep case frames.

6.3 Using high-quality deep case frames for SRL

Syntactic information such as dependencies is essential for SRL. In Section 4, we used surface

feature	description
SRFreq	the co-occurrence frequency of a predicate-argument pair without considering the semantic role of the argument
SRPmi	the PMI value for each predicate-argument pair without considering the semantic role of the argument
SRPAfreq	the frequency of a argument being a certain semantic role of a predicate
SRPami	the PMI value of an argument with its semantic role and the predicate
DCFFreq	SRFreq value calculated only from within the corresponding deep case frame
DCFPmi	SRPmi value calculated only from within the corresponding deep case frame
DCFPafreq	SRPAfreq value calculated only from within the corresponding deep case frame
DCFPami	SRPami value calculated only from within the corresponding deep case frame

Table 3: Deep case features for SRL

case frames to provide additional knowledge especially syntactic-level knowledge, for an SRL system and gained a slight improvement as shown in Section 7. Deep case frames are compiled using automatic semantic roles that use semantic-level representation. Thus, we consider that using deep case frames as additional knowledge has a more direct impact on the performance on SRL. Similar to the method described in Section 4, we also propose a set of features extracted from deep case frames which are listed in Table 3. The first four features do not concern the predicate sense. These features are similar to the predicate-argument pair features described in Section 4. The rest four features are similar to the case frame features described in Section 4. However, because the deep case frame ID is identical to the PropBank ID, no mapping processes are needed.

7 Experiments

7.1 Experimental settings

For large-scale knowledge acquisition, 40 million sentences from Chinese Gigaword 5.0 (LDC2011T13)³ were used.

For the high-quality dependency selection approach in the knowledge construction pipeline, the Stanford parser was used to apply dependency parsing. The training section of Chinese Treebank 7.0 was used to train the dependency parser and the official development section was used to train a classifier for high-quality dependency selection. Using the official evaluation section of CTB 7.0,

³We only used sentences written in simplified characters in Chinese Gigaword.

we evaluated the quality of those selected dependencies using unlabeled attachment score (UAS), which calculates the percentage of correctly identified dependency heads.

For SRL, we used the Chinese section of CoNLL-2009 shared task data (we substituted the syntactic dependencies and dependency labels produced by the Stanford parser). Automatically obtained morphological and syntactic information (the columns begin with “P”) was used. PD and AI, AC step are regarded as multi-class classification problems. We employed OPAL⁴ to solve these problems. We set the options as follows: polynomial kernel with degree 2; passive aggressive I learner; 20 iterations. The base SRL system without using additional knowledge was used as a baseline. To examine the effect of different quality of knowledge, we used different set of PASs which were extracted under different dependency selection thresholds (20%, 50%, and 100%). The official script provided on the CoNLL-2009 shared task website was used for evaluation.

For semantic role selection, similar to dependency selection, the training section of CoNLL-2009 shared task data was used to train the base SRL model. The development section in CoNLL-2009 shared task data was used to apply automatic SRL and obtain training data for the semantic role selector. We evaluated the semantic role selection approach by calculating the percentage of correctly judged semantic roles (predicate senses are not counted). For deep case frame construction, we used the Stanford parser for syntactic analysis.

⁴<http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/opal/>

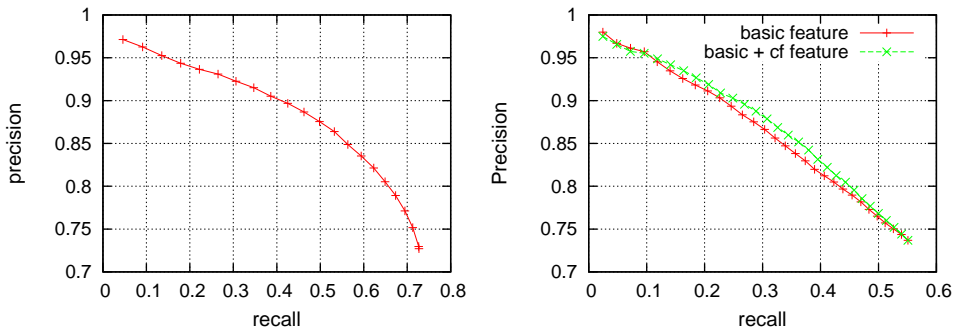


Figure 3: Precision-recall curve of dependency selection & semantic role selection

method	precision	recall	F1
baseline	80.66%	72.98%	76.63
baseline + surface case frames (100%)	79.86%	72.72%	76.12
baseline + surface case frames (50%)	80.40%	73.04%	76.54
baseline + surface case frames (20%)	80.73%	73.32%	**76.85
baseline + deep case frames (100%)	81.22%	73.55%	**77.19
baseline + deep case frames (50%)	81.30%	73.70%	**77.31
baseline + deep case frames (20%)	81.57%	73.68%	**77.42

Table 4: Evaluation results of Chinese SRL using surface and deep case frames. The ** mark and * mark mean that the result is regarded as significant (with a p value < 0.01 and a p value < 0.05, respectively) using McNemar’s test.

The base SRL system was used to assign semantic roles. We applied the proposed framework to 40 million sentences from Chinese Gigaword 5.0. We constructed deep case frames of different quality (20%, 50%, and 100%) to extract extra features to support the base SRL system.

7.2 Experimental results

Figure 3 shows the precision-recall curves of dependency selection and semantic role selection. For dependency selection, we achieved a precision over 90% when lowering the recall to around 20%. For semantic role selection, using additional surface case frame features gains a slight improvement compared to the basic features.

Table 4 shows the experimental results of SRL using surface and deep case frames as additional features. Knowledge (n%) indicates that the top n% (according to the classifier) of the automatically extracted knowledge was used. ‘100%’ means that the selection step was not applied. It is worth pointing out that when using the baseline method, we achieved an F-value of around 79.4 on CoNLL-2009 shared task original data set (where the dependency labels follow the MaltParser style, which is different from the Stanford dependencies). This result has outperformed the best sys-

tem for Chinese SRL in CoNLL-2009 shared task, which was 78.60. When applying the baseline system on the substituted version of dataset for dependency label consistency with the additional knowledge, the baseline F-value drops to 76.63. As we can see from the results, using deep case frames gained more improvements than using surface case frames. This is because deep case frames are able to directly provide semantic-level information that is insufficient in the training data of the base SRL system. Furthermore, the results show that the high-quality semantic role selection approach has a positive effect on SRL.

8 Conclusion & future work

We proposed a method for using additional knowledge to improve Chinese SRL. To address the case ambiguity problem in the surface case representation, especially for Chinese, we utilized automatic semantic roles produced by an SRL system for a better representation. The experimental results showed a promising result for high-quality semantic role selection. Also, using high-quality deep case frames that are composed of semantic roles can significantly improve the baseline SRL system.

We plan to make use of other low-level knowledge such as word embeddings (Collobert et al., 2011) and word clusters (Koo et al., 2008) to improve dependency parsing and SRL. The recent SRL approaches are mostly point-wise. Features are extracted from only pairs of the predicate and an argument candidate. We plan to design a higher-order system to capture more global features following the idea of higher-order dependency parsing. Also, reranking is widely utilized in many SRL systems and we plan to combine our surface/deep case knowledge with a reranker in order to further improve Chinese SRL.

References

- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. 3:1137–1155, February.
- Anders Björkelund, Love Hafdel, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, Colorado, June. Association for Computational Linguistics.
- Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60, Los Angeles, California, June. Association for Computational Linguistics.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. 12:2493–2537, August.
- Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Singapore, August. Association for Computational Linguistics.
- Charles J. Fillmore, Charles Wooters, and Collin F. Baker. 2001. Building a large lexical databank which provides deep semantics. In Benjamin Tsou and Olivia Kwong, editors, *Proceedings of the 15th Pacific Asia Conference on Language, Information and Computation*, Hong Kong.
- Hagen Fürstenu and Mirella Lapata. 2009. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 220–228, Athens, Greece, March. Association for Computational Linguistics.
- Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.
- Gongye Jin, Daisuke Kawahara, and Sadao Kurohashi. 2013. High quality dependency selection from automatic parses. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 947–951.
- Gongye Jin, Daisuke Kawahara, and Sadao Kurohashi. 2014. A framework for compiling high quality knowledge resources from raw corpora. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, pages 109–114.
- Gongye Jin, Daisuke Kawahara, and Sadao Kurohashi. 2015. Chinese semantic role labeling using high-quality syntactic knowledge. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 120–127, Beijing, China, July. Association for Computational Linguistics.
- Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of HLT-NAACL 2006*, pages 176–183.
- Daisuke Kawahara, Daniel Peterson, Octavian Popescu, and Martha Palmer. 2014. Inducing example-based semantic frames from a massive amount of verb uses. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–67, Gothenburg, Sweden, April. Association for Computational Linguistics.
- Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Language Resources and Evaluation*.
- Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, June. Association for Computational Linguistics.
- Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 716–724, Beijing, China, August. Coling 2010 Organizing Committee.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and*

44th Annual Meeting of the Association for Computational Linguistics, pages 337–344, Sydney, Australia, July. Association for Computational Linguistics.

Roser Morante, Vincent Van Asch, and Antal van den Bosch. 2009. Joint memory-based learning of syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 25–30, Boulder, Colorado, June. Association for Computational Linguistics.

Luiz Augusto Pizzato and Diego Mollá. 2008. Indexing on semantic roles for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 74–81, Manchester, UK, August. Coling 2008 Organizing Committee.

Buzhou Tang, Lu Li, Xinxin Li, Xuan Wang, and Xiaolong Wang. 2009. A joint syntactic and semantic dependency parsing system based on maximum entropy models. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 109–113, Boulder, Colorado, June. Association for Computational Linguistics.

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 363–373, Doha, Qatar, October. Association for Computational Linguistics.

Kun Yu, Daisuke Kawahara, and Sadao Kurohashi. 2008. Cascaded classification for high quality head-modifier pair selection. In *Proceedings of NLP 2008*, pages 1–8.

Beñat Zepirain, Eneko Agirre, and Lluís Màrquez. 2009. Generalizing over lexical features: Selectional preferences for semantic role classification. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 73–76, Suntec, Singapore, August. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 55–60, Boulder, Colorado, June. Association for Computational Linguistics.

Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities

Yadollah Yaghoobzadeh and Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
yadollah@cis.lmu.de

Abstract

Entities are essential elements of natural language. In this paper, we present methods for learning multi-level representations of entities on three complementary levels: **character** (character patterns in entity names extracted, e.g., by neural networks), **word** (embeddings of words in entity names) and **entity** (entity embeddings). We investigate state-of-the-art learning methods on each level and find large differences, e.g., for deep learning models, traditional ngram features and the subword model of *fasttext* (Bojanowski et al., 2016) on the **character** level; for *word2vec* (Mikolov et al., 2013) on the **word** level; and for the order-aware model *wang2vec* (Ling et al., 2015a) on the **entity** level.

We confirm experimentally that each level of representation contributes complementary information and a joint representation of all three levels improves the existing embedding based baseline for fine-grained entity typing by a large margin. Additionally, we show that adding information from entity descriptions further improves multi-level representations of entities.

1 Introduction

Knowledge about entities is essential for understanding human language. This knowledge can be attributional (e.g., *canFly*, *isEdible*), type-based (e.g., *isFood*, *isPolitician*, *isDisease*) or relational (e.g., *marriedTo*, *bornIn*). Knowledge bases (KBs) are designed to store this information in a structured way, so that it can be queried easily. Examples of such KBs are Freebase (Bollacker et al., 2008), Wikipedia, Google knowledge graph and

YAGO (Suchanek et al., 2007). For automatic updating and completing the entity knowledge, text resources such as news, user forums, textbooks or any other data in the form of text are important sources. Therefore, information extraction methods have been introduced to extract knowledge about entities from text. In this paper, we focus on the extraction of entity types, i.e., assigning types to – or *typing* – entities. Type information can help extraction of relations by applying constraints on relation arguments.

We address a problem setting in which the following are given: a KB with a set of entities E , a set of types T and a membership function $m : E \times T \mapsto \{0, 1\}$ such that $m(e, t) = 1$ iff entity e has type t ; and a large corpus C in which mentions of E are annotated. In this setting, we address the task of *fine-grained entity typing*: we want to learn a probability function $S(e, t)$ for a pair of entity e and type t and based on $S(e, t)$ infer whether $m(e, t) = 1$ holds, i.e., whether entity e is a member of type t .

We address this problem by learning a multi-level representation for an entity that contains the information necessary for typing it. One important source is the *contexts in which the entity is used*. We can take the standard method of learning embeddings for words and extend it to learning embeddings for entities. This requires the use of an entity linker and can be implemented by replacing all occurrences of the entity by a unique token. We refer to entity embeddings as *entity-level representations*. Previously, entity embeddings have been learned mostly using bag-of-word models like *word2vec* (e.g., by Wang et al. (2014) and Yaghoobzadeh and Schütze (2015)). We show below that order information is critical for high-quality entity embeddings.

Entity-level representations are often uninformative for rare entities, so that using only entity

embeddings is likely to produce poor results. In this paper, we use *entity names* as a source of information that is complementary to entity embeddings. We define an entity name as a noun phrase that is used to refer to an entity. We learn character and word level representations of entity names.

For the *character-level representation*, we adopt different character-level neural network architectures. Our intuition is that there is sub/cross word information, e.g., orthographic patterns, that is helpful to get better entity representations, especially for rare entities. A simple example is that a three-token sequence containing an initial like “P.” surrounded by two capitalized words (“Rolph P. Kugl”) is likely to refer to a person.

We compute the *word-level representation* as the sum of the embeddings of the words that make up the entity name. The sum of the embeddings accumulates evidence for a type/property over all constituents, e.g., a name containing “stadium”, “lake” or “cemetery” is likely to refer to a location. In this paper, we compute our word level representation with two types of word embeddings: (i) using only contextual information of words in the corpus, e.g., by `word2vec` (Mikolov et al., 2013) and (ii) using subword as well as contextual information of words, e.g., by Facebook’s recently released `fasttext` (Bojanowski et al., 2016).

In this paper, we integrate character-level and word-level with entity-level representations to improve the results of previous work on fine-grained typing of KB entities. We also show how descriptions of entities in a KB can be a complementary source of information to our multi-level representation to improve the results of entity typing, especially for rare entities.

Our main contributions in this paper are:

- We propose new methods for learning entity representations on three levels: character-level, word-level and entity-level.
- We show that these levels are complementary and a joint model that uses all three levels improves the state of the art on the task of fine-grained entity typing by a large margin.
- We experimentally show that an order dependent embedding is more informative than its bag-of-word counterpart for entity representation.

We release our dataset and source codes: cistern.cis.lmu.de/figment2/.

2 Related Work

Entity representation. Two main sources of information used for learning entity representation are: (i) links and descriptions in KB, (ii) name and contexts in corpora. We focus on name and contexts in corpora, but we also include (Wikipedia) descriptions. We represent entities on three levels: entity, word and character. Our entity-level representation is similar to work on relation extraction (Wang et al., 2014; Wang and Li, 2016), entity linking (Yamada et al., 2016; Fang et al., 2016), and entity typing (Yaghoobzadeh and Schütze, 2015). Our word-level representation with distributional word embeddings is similarly used to represent entities for entity linking (Sun et al., 2015) and relation extraction (Socher et al., 2013; Wang et al., 2014). Novel entity representation methods we introduce in this paper are representation based on `fasttext` (Bojanowski et al., 2016) subword embeddings, several character-level representations, “order-aware” entity-level embeddings and the combination of several different representations into one multi-level representation.

Character-subword level neural networks. Character-level convolutional neural networks (CNNs) are applied by dos Santos and Zadrozny (2014) to part of speech (POS) tagging, by dos Santos and Guimarães (2015), Ma and Hovy (2016), and Chiu and Nichols (2016) to named entity recognition (NER), by Zhang et al. (2015) and Zhang and LeCun (2015) to sentiment analysis and text categorization, and by Kim et al. (2016) to language modeling (LM). Character-level LSTM is applied by Ling et al. (2015b) to LM and POS tagging, by Lample et al. (2016) to NER, by Ballesteros et al. (2015) to parsing morphologically rich languages, and by Cao and Rei (2016) to learning word embeddings. Bojanowski et al. (2016) learn word embeddings by representing words with the average of their character ngrams (subwords) embeddings. Similarly, Chen et al. (2015) extends `word2vec` for Chinese with joint modeling with characters.

Fine-grained entity typing. Our task is to infer fine-grained types of KB entities. KB completion is an application of this task. Yaghoobzadeh and Schütze (2015)’s FIGMENT system addresses this task with only contextual information; they do not use character-level and word-level features of entity names. Neelakantan and Chang (2015) and Xie et al. (2016) also address a similar task,

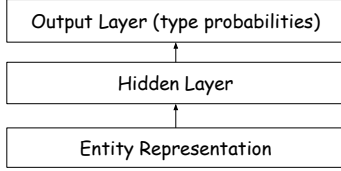


Figure 1: Schematic diagram of our architecture for entity classification. “Entity Representation” ($\vec{v}(e)$) is the (one-level or multi-level) vector representation of entity. Size of output layer is $|T|$.

but they rely on entity descriptions in KBs, which in many settings are not available. The problem of Fine-grained mention typing (FGMT) (Yosef et al., 2012; Ling and Weld, 2012; Yogatama et al., 2015; Del Corro et al., 2015; Shimaoka et al., 2016; Ren et al., 2016) is related to our task. FGMT classifies single *mentions* of named entities to their context dependent types whereas we attempt to identify all types of a KB *entity* from the aggregation of all its mentions. FGMT can still be evaluated in our task by aggregating the mention level decisions but as we will show in our experiments for one system, i.e., FIGER (Ling and Weld, 2012), our entity embedding based models are better in entity typing.

3 Fine-grained entity typing

Given (i) a KB with a set of entities E , (ii) a set of types T , and (iii) a large corpus C in which mentions of E are linked, we address the task of *fine-grained entity typing* (Yaghoobzadeh and Schütze, 2015): predict whether entity e is a member of type t or not. To do so, we use a set of training examples to learn $P(t|e)$: the probability that entity e has type t . These probabilities can be used to assign *new types* to entities covered in the KB as well as typing *unknown entities*.

We learn $P(t|e)$ with a general architecture; see Figure 1. The output layer has size $|T|$. Unit t of this layer outputs the probability for type t . “Entity Representation” ($\vec{v}(e)$) is the vector representation of entity e – we will describe in detail in the rest of this section what forms $\vec{v}(e)$ takes. We model $P(t|e)$ as a multi-label classification, and train a multilayer perceptron (MLP) with one hidden layer:

$$[P(t_1|e) \dots P(t_T|e)] = \sigma(\mathbf{W}_{\text{out}} f(\mathbf{W}_{\text{in}} \vec{v}(e))) \quad (1)$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{h \times d}$ is the weight matrix from

$\vec{v}(e) \in \mathbb{R}^d$ to the hidden layer with size h . f is the rectifier function. $\mathbf{W}_{\text{out}} \in \mathbb{R}^{|T| \times h}$ is the weight matrix from hidden layer to output layer of size $|T|$. σ is the sigmoid function. Our objective is binary cross entropy summed over types:

$$\sum_t - (m_t \log p_t + (1 - m_t) \log (1 - p_t))$$

where m_t is the truth and p_t the prediction.

The key difficulty when trying to compute $P(t|e)$ is in learning a good representation for entity e . We make use of contexts and name of e to represent its feature vector on the three levels of entity, word and character.

3.1 Entity-level representation

Distributional representations or embeddings are commonly used for words. The underlying hypothesis is that words with similar meanings tend to occur in similar contexts (Harris, 1954) and therefore cooccur with similar context words. We can extend the distributional hypothesis to entities (cf. Wang et al. (2014), Yaghoobzadeh and Schütze (2015)): entities with similar meanings tend to have similar contexts. Thus, we can learn a d dimensional embedding $\vec{v}(e)$ of entity e from a corpus in which all mentions of the entity have been replaced by a special identifier. We refer to these entity vectors as the *entity level representation* (ELR).

In previous work, order information of context words (relative position of words in the contexts) was generally ignored and objectives similar to the SkipGram (henceforth: *SKIP*) model were used to learn $\vec{v}(e)$. However, the bag-of-words context is difficult to distinguish for pairs of types like (restaurant,food) and (author,book). This suggests that *using order aware embedding models is important for entities*. Therefore, we apply Ling et al. (2015a)’s extended version of SKIP, Structured SKIP (SSKIP). It incorporates the order of context words into the objective. We compare it with SKIP embeddings in our experiments.

3.2 Word-level representation

Words inside entity names are important sources of information for typing entities. We define the word-level representation (WLR) as the *average of the embeddings of the words* that the entity name contains $\vec{v}(e) = 1/n \sum_{i=1}^n \vec{v}(w_i)$ where $\vec{v}(w_i)$ is the embedding of the i^{th} word of an entity name

of length n . We opt for simple averaging since entity names often consist of a small number of words with clear semantics. Thus, averaging is a promising way of combining the information that each word contributes.

The word embedding, \vec{w} , itself can be learned from models with different granularity levels. Embedding models that consider words as atomic units in the corpus, e.g., SKIP and SSKIP, are word-level. On the other hand, embedding models that represent words with their character ngrams, e.g., *fasttext* (Bojanowski et al., 2016), are subword-level. Based on this, we consider and evaluate **word-level WLR (WWLR)** and **subword-level WLR (SWLR)** in this paper.¹

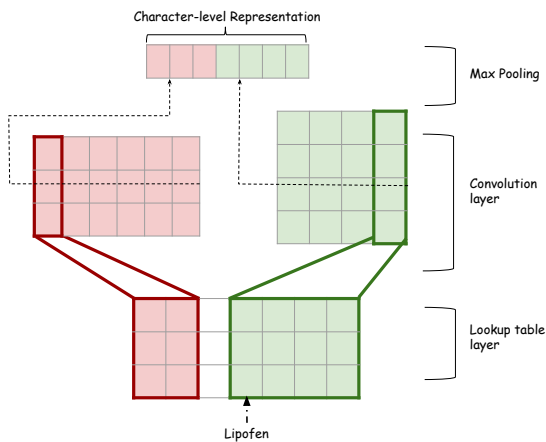


Figure 2: Example architecture for the character-level CNN with max pooling. The input is “Lipofen”. Character embedding size is three. There are three filters of width 2 and four filters of width 4.

3.3 Character-level representation

For computing the *character level representation* (CLR), we design models that try to type an entity based on the sequence of characters of its name. Our hypothesis is that names of entities of a specific type often have similar character patterns. Entities of type ETHNICITY often end in “ish” and “ian”, e.g., “Spanish” and “Russian”. Entities of type MEDICINE often end in “en”: “Lipofen”, “acetaminophen”. Also, some types tend to have specific cross-word shapes in their entities, e.g.,

¹Subword models have properties of both character-level models (subwords are character ngrams) and of word-level models (they do not cross boundaries between words). They probably could be put in either category, but in our context fit the word-level category better because we see the granularity level with respect to the entities and not words.

PERSON names usually consist of two words, or MUSIC names are usually long, containing several words.

The first layer of the character-level models is a *lookup table* that maps each character to an embedding of size d_c . These embeddings capture similarities between characters, e.g., similarity in type of phoneme encoded (consonant/vowel) or similarity in case (lower/upper). The output of the lookup layer for an entity name is a matrix $C \in \mathbb{R}^{l \times d_c}$ where l is the maximum length of a name and all names are padded to length l . This length l includes special start/end characters that bracket the entity name.

We experiment with four architectures to produce character-level representations in this paper: FORWARD (direct forwarding of character embeddings), CNNs, LSTMs and BiLSTMs. The output of each architecture then takes the place of the entity representation $\vec{v}(e)$ in Figure 1.

FORWARD simply concatenates all rows of matrix C ; thus, $\vec{v}(e) \in \mathbb{R}^{d_c * l}$.

The **CNN** uses k filters of different window widths w to narrowly convolve C . For each filter $H \in \mathbb{R}^{d_c \times w}$, the result of the convolution of H over matrix C is feature map $f \in \mathbb{R}^{l-w+1}$:

$$f[i] = \text{rectifier}(C_{[:,i:i+w-1]} \odot H + b)$$

where *rectifier* is the activation function, b is the bias, $C_{[:,i:i+w-1]}$ are the columns i to $i + w - 1$ of C , $1 \leq w \leq 10$ are the window widths we consider and \odot is the sum of element-wise multiplication. Max pooling then gives us one feature for each filter. The concatenation of all these features is our representation: $\vec{v}(e) \in \mathbb{R}^k$. An example CNN architecture is show in Figure 2.

The input to the **LSTM** is the character sequence in matrix C , i.e., $x_1, \dots, x_l \in \mathbb{R}^{d_c}$. It generates the state sequence h_1, \dots, h_{l+1} and the output is the last state $\vec{v}(e) \in \mathbb{R}^{d_h}$.²

The **BiLSTM** consists of two LSTMs, one going forward, one going backward. The first state of the backward LSTM is initialized as h_{l+1} , the last state of the forward LSTM. The BiLSTM entity representation is the concatenation of last states of forward and backward LSTMs, i.e., $\vec{v}(e) \in \mathbb{R}^{2 * d_h}$.

3.4 Multi-level representations

Our different levels of representations can give complementary information about entities.

²We use Blocks (van Merriënboer et al., 2015).

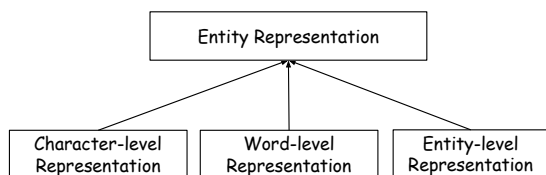


Figure 3: Multi-level representation

WLR and CLR. Both WLR models, SWLR and WWLR, do not have access to the cross-word character ngrams of entity names while CLR models do. Also, CLR is task specific by training on the entity typing dataset while WLR is generic. On the other hand, WWLR and SWLR models have access to information that CLR ignores: the tokenization of entity names into words and embeddings of these words. It is clear that words are particularly important character sequences since they often correspond to linguistic units with clearly identifiable semantics – which is not true for most character sequences. For many entities, the words they contain are a better basis for typing than the character sequence. For example, even if “nectarine” and “compote” did not occur in any names in the training corpus, we can still learn good word embeddings from their non-entity occurrences. This then allows us to correctly type the entity “Aunt Mary’s Nectarine Compote” as FOOD based on the sum of the word embeddings.

WLR/CLR and ELR. Representations from entity names, i.e., WLR and CLR, by themselves are limited because many classes of names can be used for different types of entities; e.g., person names do not contain hints as to whether they are referring to a politician or athlete. In contrast, the ELR embedding is based on an entity’s contexts, which are often informative for each entity and can distinguish politicians from athletes. On the other hand, not all entities have sufficiently many informative contexts in the corpus. For these entities, their name can be a complementary source of information and character/word level representations can increase typing accuracy.

Thus, we introduce joint models that use combinations of the three levels. Each multi-level model concatenates several levels. We train the constituent embeddings as follows. WLR and ELR are computed as described above and are not changed during training. CLR – produced by one of the character-level networks described above – is initialized randomly and then tuned during

training. Thus, it can focus on complementary information related to the task that is not already present in other levels. The schematic diagram of our multi-level representation is shown in Figure 3.

4 Experimental setup and results

4.1 Setup

Entity datasets and corpus. We address the task of fine-grained entity typing and use Yaghoobzadeh and Schütze (2015)’s FIGMENT dataset³ for evaluation. The FIGMENT corpus is part of a version of ClueWeb in which Freebase entities are annotated using FACC1 (URL, 2016b; Gabrilovich et al., 2013). The FIGMENT entity datasets contain 200,000 Freebase entities that were mapped to 102 FIGER types (Ling and Weld, 2012). We use the same train (50%), dev (20%) and test (30%) partitions as Yaghoobzadeh and Schütze (2015) and extract the names from mentions of dataset entities in the corpus. We take the most frequent name for dev and test entities and three most frequent names for train (each one tagged with entity types).

Adding parent types to refine entity dataset. FIGMENT ignores that FIGER is a proper hierarchy of types; e.g., while HOSPITAL is a subtype of BUILDING according to FIGER, there are entities in FIGMENT that are hospitals, but not buildings.⁴ Therefore, we modified the FIGMENT dataset by adding for each assigned type (e.g., HOSPITAL) its parents (e.g., BUILDING). This makes FIGMENT more consistent and eliminates spurious false negatives (BUILDING in the example).

We now describe our **baselines**: (i) BOW & NSL: hand-crafted features, (ii) FIGMENT (Yaghoobzadeh and Schütze, 2015) and (iii) adapted version of FIGER (Ling and Weld, 2012).

We implement the following two feature sets from the literature as a *hand-crafted baseline* for our character and word level models. (i) *BOW*: individual words of entity name (both as-is and lowercased); (ii) *NSL* (ngram-shape-length): shape and length of the entity name (cf. Ling and Weld (2012)), character n -grams, $1 \leq n \leq n_{\max}$, $n_{\max} = 5$ (we also tried $n_{\max} = 7$, but results were worse on dev) and normalized character n -grams: lowercased, digits replaced by “7”, punctuation replaced by “.”. These features are represented as a sparse

³cistern.cis.lmu.de/figment/

⁴See github.com/xiaoling/figer for FIGER

binary vector $\vec{v}(e)$ that is input to the architecture in Figure 1.

FIGMENT is the model for entity typing presented by Yaghoobzadeh and Schütze (2015). The authors only use entity-level representations for entities trained by SkipGram, so the *FIGMENT* baseline corresponds to the entity-level result shown as ELR(SKIP) in the tables.

The third baseline is using an existing mention-level entity typing system, *FIGER* (Ling and Weld, 2012). *FIGER* uses a wide variety of features on different levels (including parsing-based features) from contexts of entity mentions as well as the mentions themselves and returns a score for each mention-type instance in the corpus. We provide the ClueWeb/FACC1 segmentation of entities, so *FIGER* does not need to recognize entities.⁵ We use the trained model provided by the authors and normalize *FIGER* scores using softmax to make them comparable for aggregation. We experimented with different aggregation functions (including maximum and k-largest-scores for a type), but we use the average of scores since it gave us the best result on dev. We call this baseline AGG-*FIGER*.

Distributional embeddings. For WWLR and ELR, we use SkipGram model in `word2vec` and SSkip model in `wang2vec` (Ling et al., 2015a) to learn embeddings for words, entities and types. To obtain embeddings for all three in the same space, we process ClueWeb/FACC1 as follows. For each sentence s , we add three copies: s itself, a copy of s in which each entity is replaced with its Freebase identifier (MID) and a copy in which each entity (not test entities though) is replaced with an ID indicating its notable type. The resulting corpus contains around 4 billion tokens and 1.5 billion types.

We run SKIP and SSkip with the same setup (200 dimensions, 10 negative samples, window size 5, word frequency threshold of 100)⁶ on this corpus to learn embeddings for words, entities and *FIGER* types. Having entities and types in the same vector space, we can add another feature vector $\vec{v}(e) \in \mathbb{R}^{|T|}$ (referred to as TC below): for each entity, we compute cosine similarity of its entity vector with all type vectors.

For SWLR, we use `fasttext`⁷ to learn word

⁵Mention typing is separated from recognition in *FIGER* model. So it can use our segmentation of entities.

⁶The threshold does not apply for MIDs.

⁷github.com/facebookresearch/fastText

embeddings from the ClueWeb/FACC1 corpus. We use similar settings as our WWLR SKIP and SSkip embeddings and keep the defaults of other hyperparameters. Since the trained model of `fasttext` is applicable for new words, we apply the model to get embeddings for the filtered rare words as well.

model	hyperparameters
CLR(FF)	$d_c = 15, h_{mlp} = 600$
CLR(LSTM)	$d_c = 70, d_h = 70, h_{mlp} = 300$
CLR(BiLSTM)	$d_c = 50, d_h = 50, h_{mlp} = 200$
CLR(CNN)	$d_c = 10, w = [1, \dots, 8]$ $n = 100, h_{mlp} = 800$
CLR(NSL)	$h_{mlp} = 800$
BOW	$h_{mlp} = 200$
BOW+CLR(NSL)	$h_{mlp} = 300$
WWLR	$h_{mlp} = 400$
SWLR	$h_{mlp} = 400$
WWLR+CLR(CNN)	$w = [1, \dots, 7]$ $d_c = 10, n = 50, h_{mlp} = 700$
SWLR+CLR(CNN)	$w = [1, \dots, 7]$ $d_c = 10, n = 50, h_{mlp} = 700$
ELR(SKIP)	$h_{mlp} = 400$
ELR(SSkip)	$h_{mlp} = 400$
ELR+CLR	$d_c = 10, w = [1, \dots, 7]$ $n = 100, h_{mlp} = 700$
ELR+WWLR	$h_{mlp} = 600$
ELR+SWLR	$h_{mlp} = 600$
ELR+WWLR+CLR	$d_c = 10, w = [1, \dots, 7]$ $n = 50, h_{mlp} = 700$
ELR+SWLR+CLR	$d_c = 10, w = [1, \dots, 7]$ $n = 50, h_{mlp} = 700$
ELR+WWLR+CNN+TC	$d_c = 10, w = [1, \dots, 7]$ $n = 50, h_{mlp} = 900$
ELR+SWLR+CNN+TC(MuLR)	$d_c = 10, w = [1, \dots, 7]$ $n = 50, h_{mlp} = 900$
AVG-DES	$h_{mlp} = 400$
MuLR+AVG-DES	$d_c = 10, w = [1, \dots, 7]$ $n = 50, h_{mlp} = 1000$

Table 1: Hyperparameters of different models. w is the filter size. n is the number of CNN feature maps for each filter size. d_c is the character embedding size. d_h is the LSTM hidden state size. h_{mlp} is the number of hidden units in the MLP.

Our **hyperparameter values** are given in Table 1. The values are optimized on dev. We use AdaGrad and minibatch training. For each experiment, we select the best model on dev.

We use these **evaluation measures**: (i) accuracy: an entity is correct if all its types and no incorrect types are assigned to it; (ii) micro average F_1 : F_1 of all type-entity assignment decisions; (iii) entity macro average F_1 : F_1 of types assigned to an entity, averaged over entities; (iv) type macro average F_1 : F_1 of entities assigned to a type, averaged over types.

The assignment decision is based on thresholding the probability function $P(t|e)$. For each model and type, we select the threshold that maximizes F_1 of entities assigned to the type on dev.

	all entities			head entities			tail entities		
	acc	mic	mac	acc	mic	mac	acc	mic	mac
1 MFT	.000	.041	.041	.000	.044	.044	.000	.038	.038
2 CLR(FORWARD)	.066	.379	.352	.067	.342	.369	.061	.374	.350
3 CLR(LSTM)	.121	.425	.396	.122	.433	.390	.116	.408	.391
4 CLR(BiLSTM)	.133	.440	.404	.129	.443	.394	.135	.428	.404
5 CLR(NSL)	.164	.484	.464	.157	.470	.443	.173	.483	.472
6 CLR(CNN)	.177	.494	.468	.171	.484	.450	.187	.489	.474
7 BOW	.113	.346	.379	.109	.323	.353	.120	.356	.396
8 WWLR(SKIP)	.214	.581	.531	.293	.660	.634	.173	.528	.478
9 WWLR(SSKIP)	.223	.584	.543	.306	.667	.642	.183	.533	.494
10 SWLR	.236	.590	.554	.301	.665	.632	.209	.551	.522
11 BOW+CLR(NSL)	.156	.487	.464	.157	.480	.452	.159	.485	.469
12 WWLR+CLR(CNN)	.257	.603	.568	.317	.668	.637	.235	.567	.538
13 SWLR+CLR(CNN)	.241	.594	.561	.295	.659	.628	.227	.560	.536
14 ELR(SKIP)	.488	.774	.741	.551	.834	.815	.337	.621	.560
15 ELR(SSKIP)	.515	.796	.763	.560	.839	.819	.394	.677	.619
16 AGG-FIGER	.320	.694	.660	.396	.762	.724	.220	.593	.568
17 ELR+CLR	.554	.816	.788	.580	.844	.825	.467	.733	.690
18 ELR+WWLR	.557	.819	.793	.582	.846	.827	.480	.749	.708
19 ELR+SWLR	.558	.820	.796	.584	.846	.829	.480	.751	.714
20 ELR+WWLR+CLR	.568	.823	.798	.590	.847	.829	.491	.755	.716
21 ELR+SWLR+CLR	.569	.824	.801	.590	.849	.831	.497	.760	.724
22 ELR+WWLR+CLR+TC	.572	.824	.801	.594	.849	.831	.499	.759	.722
23 ELR+SWLR+CLR+TC	.575	.826	.802	.597	.851	.831	.508	.762	.727

Table 2: Accuracy (acc), micro (mic) and macro (mac) F_1 on test for all, head and tail entities.

4.2 Results

Table 2 gives results on the test entities for all (about 60,000 entities), head (frequency > 100; about 12,200) and tail (frequency < 5; about 10,000). *MFT* (line 1) is the most frequent type baseline that ranks types according to their frequency in the train entities. Each level of representation is separated with dashed lines, and – unless noted otherwise – the best of each level is joined in multi level representations.⁸

Character-level models are on lines 2-6. The order of systems is: CNN > NSL > BiLSTM > LSTM > FORWARD. The results show that complex neural networks are more effective than simple forwarding. BiLSTM works better than LSTM, confirming other related work. CNNs probably work better than LSTMs because there are few complex non-local dependencies in the sequence, but many important local features. CNNs with maxpooling can more straightforwardly capture local and position-independent features. CNN also beats NSL baseline; a possible reason is that CNN – an automatic method of feature learning

⁸For accuracy measure: in the following ordered lists of sets, $A < B$ means that all members (row numbers in Table 2) of A are significantly worse than all members of B : $\{1\} < \{2\} < \{3, \dots, 11\} < \{12, 13\} < \{14, 15, 16\} < \{17, \dots, 23\}$. Test of equal proportions, $\alpha < 0.05$. See Table 6 in the appendix for more details.

types:	all	head	tail
AGG-FIGER	.566	.702	.438
ELR	.621	.784	.480
MuLR	.669	.811	.541

Table 3: Type macro average F_1 on test for all, head and tail types. MuLR = ELR+SWLR+CLR+TC

	all	known?	
	yes	no	
CLR(NSL)	.484	.521	.341
CLR(CNN)	.494	.524	.374
BOW	.346	.435	.065
SWLR	.590	.612	.499
BOW+NSL	.497	.535	.358
SWLR+CLR(CNN)	.594	.616	.508

Table 4: Micro F_1 on test of character, word level models for all, known (“known? yes”) and unknown (“known? no”) entities.

– is more robust than hand engineered feature based NSL. We show more detailed results in Section 4.3.

Word-level models are on lines 7-10. BOW performs worse than WWLR because it cannot deal well with sparseness. SSKIP uses word order information in WWLR and performs better than SKIP. SWLR uses subword information and performs better than WWLR, especially for tail entities. Integrating subword information improves the quality of embeddings for rare words and mitigates the problem of unknown words.

Joint word-character level models are on lines 11-13. WWLR+CLR(CNN) and SWLR+CLR(CNN) beat the component models. This confirms our underlying assumption in designing the complementary multi-level models. BOW problem with rare words does not allow its joint model with NSL to work better than NSL. WWLR+CLR(CNN) works better than BOW+CLR(NSL) by 10% micro F_1 , again due to the limits of BOW compared to WWLR. Interestingly WWLR+CLR works better than SWLR+CLR and this suggests that WWLR is indeed richer than SWLR when CLR mitigates its problem with rare/unknown words

Entity-level models are on lines 14–15 and they are better than all previous models on lines

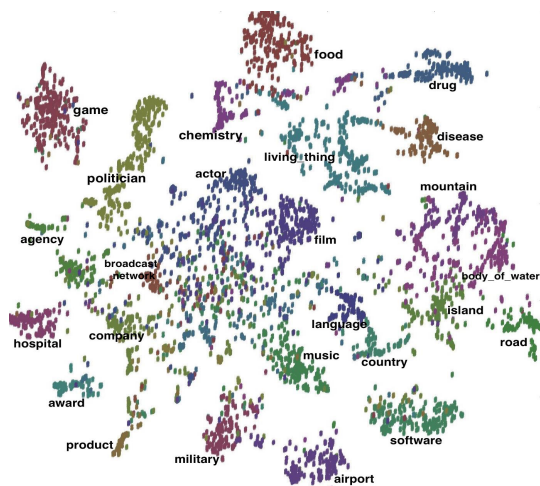


Figure 4: t-SNE result of entity-level representations

1–13. This shows the power of entity-level embeddings. In Figure 4, a t-SNE (Van der Maaten and Hinton, 2008) visualization of ELR(SKIP) embeddings using different colors for entity types shows that entities of the same type are clustered together. SSKIP works marginally better than SKIP for ELR, especially for tail entities, confirming our hypothesis that order information is important for a good distributional entity representation. This is also confirming the results of Yaghoobzadeh and Schütze (2016), where they also get better entity typing results with SSKIP compared to SKIP. They propose to use entity typing as an extrinsic evaluation for embedding models.

Joint entity, word, and character level models are on lines 16-23. The AGG-FIGER baseline works better than the systems on lines 1-13, but worse than ELRs. This is probably due to the fact that AGG-FIGER is optimized for mention typing and it is trained using distant supervision assumption. Parallel to our work, Yaghoobzadeh et al. (2017) optimize a mention typing model for our entity typing task by introducing multi instance learning algorithms, resulting comparable performance to ELR(SKIP). We will investigate their method in future.

Joining CLR with ELR (line 17) results in large improvements, especially for tail entities (5% micro F_1). This demonstrates that for rare entities, contextual information is often not sufficient for an informative representation, hence name features are important. This is also true for the joint models of WWLR/SWLR and ELR (lines 18-19). Joining WWLR works better than

CLR, and SWLR is slightly better than WWLR. Joint models of WWLR/SWLR with ELR+CLR gives more improvements, and SWLR is again slightly better than WWLR. ELR+WWLR+CLR and ELR+SWLR+CLR, are better than their two-level counterparts, again confirming that these levels are complementary.

We get a further boost, especially for tail entities, by also including TC (type cosine) in the combinations (lines 22-23). This demonstrates the potential advantage of having a common representation space for entities and types. Our best model, ELR+SWLR+CLR+TC (line 22), which we refer to as MuLR in the other tables, beats our initial baselines (ELR and AGG-FIGER) by large margins, e.g., in tail entities improvements are more than 8% in micro F1.

Table 3 shows **type macro F_1** for MuLR (ELR+SWLR+CLR+TC) and two baselines. There are 11 head types (those with ≥ 3000 train entities) and 36 tail types (those with < 200 train entities). These results again confirm the superiority of our multi-level models over the baselines: AGG-FIGER and ELR, the best single-level model baseline.

4.3 Analysis

Unknown vs. known entities. To analyze the complementarity of character and word level representations, as well as more fine-grained comparison of our models and the baselines, we divide test entities into *known entities* – at least one word of the entity’s name appears in a train entity – and *unknown entities* (the complement). There are 45,000 (resp. 15,000) known (resp. unknown) test entities.

Table 4 shows that the CNN works only slightly better (by 0.3%) than NSL on known entities, but works much better on unknown entities (by 3.3%), justifying our preference for deep learning CLR models. As expected, BOW works relatively well for known entities and really poorly for unknown entities. SWLR beats CLR models as well as BOW. The reason is that in our setup, word embeddings are induced on the entire corpus using an unsupervised algorithm. Thus, even for many words that did not occur in train, SWLR has access to informative representations of words. The joint model, SWLR+CLR(CNN), is significantly better than BOW+CLR(NSL) again due to limits of BOW. SWLR+CLR(CNN) is better than SWLR

in unknown entities.

Case study of LIVING-THING. To understand the interplay of different levels better, we perform a case study of the type LIVING-THING. Living beings that are not humans belong to this type.

WLRs incorrectly assign “Walter Leaf” (PERSON) and “Along Came A Spider” (MUSIC) to LIVING-THING because these names contain a word referring to a LIVING-THING (“leaf”, “spider”), but the entity itself is not a LIVING-THING. In these cases, the averaging of embeddings that WLR performs is misleading. The CLR(CNN) types these two entities correctly because their names contain character ngram/shape patterns that are indicative of PERSON and MUSIC.

ELR incorrectly assigns “Zumpango” (CITY) and “Lake Kasumigaura” (LOCATION) to LIVING-THING because these entities are rare and words associated with living things (e.g., “wildlife”) dominate in their contexts. However, CLR(CNN) and WLR enable the joint model to type the two entities correctly: “Zumpango” because of the informative suffix “-go” and “Lake Kasumigaura” because of the informative word “Lake”.

While some of the **remaining errors** of our best system MuLR are due to the inherent difficulty of entity typing (e.g., it is difficult to correctly type a one-word entity that occurs once and whose name is not informative), many other errors are due to artifacts of our setup. First, ClueWeb/FACCI is the result of an automatic entity linking system and any entity linking errors propagate to our models. Second, due to the incompleteness of Freebase (Yaghoobzadeh and Schütze, 2015), many entities in the FIGMENT dataset are incompletely annotated, resulting in correctly typed entities being evaluated as incorrect.

Adding another source: description-based embeddings. While in this paper, we focus on the contexts and names of entities, there is a textual source of information about entities in KBs which we can also make use of: descriptions of entities. We extract Wikipedia descriptions of FIGMENT entities filtering out the entities ($\sim 40,000$ out of $\sim 200,000$) without description.

We then build a simple entity representation by averaging the embeddings of the top k words (wrt tf-idf) of the description (henceforth, AVG-DES).⁹ This representation is used as input in Figure 1 to train the MLP. We also train our best multi-

⁹ $k = 20$ gives the best results on dev.

entities:	all	head	tail
AVG-DES	.773	.791	.745
MuLR	.825	.846	.757
MuLR+AVG-DES	.873	.877	.852

Table 5: Micro average F_1 results of MuLR and description based model and their joint.

level model as well as the joint of the two on this smaller dataset. Since the descriptions are coming from Wikipedia, we use 300-dimensional Glove (URL, 2016a) embeddings pretrained on Wikipedia+Gigaword to get more coverage of words. For MuLR, we still use the embeddings we trained before.

Results are shown in Table 5. While for head entities, MuLR works marginally better, the difference is very small in tail entities. The joint model of the two (by concatenation of vectors) improves the micro F1, with clear boost for tail entities. This suggests that for tail entities, the contextual and name information is not enough by itself and some keywords from descriptions can be really helpful. Integrating more complex description-based embeddings, e.g., by using CNN (Xie et al., 2016), may improve the results further. We leave it for future work.

5 Conclusion

In this paper, we have introduced representations of entities on different levels: character, word and entity. The character level representation is learned from the entity name. The word level representation is computed from the embeddings of the words w_i in the entity name where the embedding of w_i is derived from the corpus contexts of w_i . The entity level representation of entity e_i is derived from the corpus contexts of e_i . Our experiments show that each of these levels contributes complementary information for the task of fine-grained typing of entities. The joint model of all three levels beats the state-of-the-art baseline by large margins. We further showed that extracting some keywords from Wikipedia descriptions of entities, when available, can considerably improve entity representations, especially for rare entities. We believe that our findings can be transferred to other tasks where entity representation matters.

Acknowledgments. This work was supported by DFG (SCHU 2246/8-2).

References

- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.
- Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.
- Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 18–26, Berlin, Germany, August. Association for Computational Linguistics.
- Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1236–1242.
- Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.
- Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878, Lisbon, Portugal, September. Association for Computational Linguistics.
- Cícero Nogueira dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. *CoRR*, abs/1505.05008.
- Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1818–1826.
- Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 260–269, Berlin, Germany, August. Association for Computational Linguistics.
- Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of cluweb corpora.
- Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.
- Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*.
- Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado, May–June. Association for Computational Linguistics.
- Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fernandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September. Association for Computational Linguistics.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525, Denver, Colorado, May–June. Association for Computational Linguistics.

- Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1825–1834.
- Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. pages 69–74, June.
- Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 926–934.
- Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706.
- Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1333–1339.
- URL. 2016a. Glove project. <http://nlp.stanford.edu/projects/glove>.
- URL. 2016b. Lemur project. <http://lemurproject.org/clueweb12/FACC1>.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.
- Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619.
- Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1293–1299.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar, October. Association for Computational Linguistics.
- Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2659–2665.
- Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, Lisbon, Portugal, September. Association for Computational Linguistics.
- Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246, Berlin, Germany, August. Association for Computational Linguistics.
- Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *EACL*, Valencia, Spain.
- Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. pages 250–259, August.
- Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 291–296, Beijing, China, July. Association for Computational Linguistics.
- Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: hierarchical type classification for entity names. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 1361–1370.
- Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.
- Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. pages 649–657.

A Supplementary Material

All entities		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01	MFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	CLR(FORWARD)	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	CLR(LSTM)	*	*	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	CLR(BiLSTM)	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	CLR(CNN)	*	*	*	*	0	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
06	CLR(NSL)	*	*	*	*	0	0	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
07	BOW	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	WWLR(SkipG)	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
09	WWLR(SSkipG)	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
10	SWLR	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
11	BOW+CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0	0
13	SWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0
14	ELR(SkipG)	*	*	*	*	*	*	*	*	*	*	*	0	*	0	0	*	0	0	0	0	0	0	0
15	ELR(SSkipG)	*	*	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0
16	AGG-FIGER	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0
17	ELR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
18	ELR+WWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
19	ELR+SWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
20	ELR+WWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
21	ELR+SWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
22	ELR+WWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
23	ELR+SWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Head entities		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01	MFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	CLR(FORWARD)	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	CLR(LSTM)	*	*	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	CLR(BiLSTM)	*	*	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	CLR(CNN)	*	*	*	*	0	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
06	CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	BOW	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	WWLR(SkipG)	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
09	WWLR(SSkipG)	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
10	SWLR	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
11	BOW+CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0	0
13	SWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0
14	ELR(SkipG)	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0
15	ELR(SSkipG)	*	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0
16	AGG-FIGER	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0
17	ELR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
18	ELR+WWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
19	ELR+SWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
20	ELR+WWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
21	ELR+SWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
22	ELR+WWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
23	ELR+SWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Tail entities		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
01	MFT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	CLR(FORWARD)	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	CLR(LSTM)	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	CLR(BiLSTM)	*	*	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	CLR(CNN)	*	*	*	*	0	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
06	CLR(NSL)	*	*	*	*	0	0	*	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0
07	BOW	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08	WWLR(SkipG)	*	*	*	*	0	0	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
09	WWLR(SSkipG)	*	*	*	*	0	0	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
10	SWLR	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0
11	BOW+CLR(NSL)	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	WWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0
13	SWLR+CLR(CNN)	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0
14	ELR(SkipG)	*	*	*	*	*	*	*	*	*	*	*	*	0	0	*	0	0	0	0	0	0	0	0
15	ELR(SSkipG)	*	*	*	*	*	*	*	*	*	*	*	*	*	0	*	0	0	0	0	0	0	0	0
16	AGG-FIGER	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0
17	ELR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
18	ELR+WWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
19	ELR+SWLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
20	ELR+WWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
21	ELR+SWLR+CLR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
22	ELR+WWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
23	ELR+SWLR+CLR+TC	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Table 6: Significance-test results for accuracy measure for all, head and tail entities. If the result for the model in a row is significantly larger than the result for the model in a column, then the value in the corresponding (row,column) is * and otherwise is 0.

The ContrastMedium Algorithm: Taxonomy Induction From Noisy Knowledge Graphs With Just a Few Links

Stefano Faralli¹, Alexander Panchenko², Chris Biemann² and Simone Paolo Ponzetto¹

¹Data and Web Science Group, University of Mannheim, Germany

²Language Technology Group, University of Hamburg, Germany

{stefano,simone}@informatik.uni-mannheim.de

{panchenko,biemann}@informatik.uni-hamburg.de

Abstract

In this paper, we present ContrastMedium, an algorithm that transforms noisy semantic networks into full-fledged, clean taxonomies. ContrastMedium is able to identify the embedded taxonomy structure from a noisy knowledge graph without explicit human supervision such as, for instance, a set of manually selected input root and leaf concepts. This is achieved by leveraging structural information from a companion reference taxonomy, to which the input knowledge graph is linked (either automatically or manually). When used in conjunction with methods for hypernym acquisition and knowledge base linking, our methodology provides a complete solution for end-to-end taxonomy induction. We conduct experiments using automatically acquired knowledge graphs, as well as a SemEval benchmark, and show that our method is able to achieve high performance on the task of taxonomy induction.

1 Introduction

Recent years have witnessed an impressive amount of work on automatic construction of wide-coverage knowledge resources. Web-scale open information extraction systems like NELL (Carlson et al., 2010) or ReVerb (Fader et al., 2011) have been successful in acquiring massive amounts of machine-readable knowledge by effectively tapping large amounts of text from Web pages. However, the output of these systems does not consist of a clean, fully-semanticized output. Such output, on the other hand, could be provided by the vocabulary of large-scale ontologies like DBpedia (Bizer et al., 2009) or YAGO (Hoffart et al., 2013) and the integration of open and

closed information extraction approaches (Dutta et al., 2014). The use of an encyclopedia-centric (e.g., Wikipedia-based) dictionary of entities leads to poor coverage of domain-specific terminologies (Faralli and Navigli, 2013). This can be alleviated by constructing knowledge bases of ever increasing coverage and complexity from the Web (Wu et al., 2012; Gupta et al., 2014; Dong et al., 2014) or by community efforts (Bollacker et al., 2008). However, the focus on large size and wide coverage at entity level has led all these resources to avoid the complementary problem of curating and maintaining a clean taxonomic backbone with as minimal supervision as possible. That is, no resource, to date, integrates structured information from existing wide-coverage knowledge graphs with empirical evidence from text for the explicit goal of building full-fledged taxonomies consisting of a clean and fully-connected directed acyclic graph (DAG). This is despite the fact that taxonomies have been known for a long time to provide valid tools to represent domain-specific knowledge with dozens of scientific, industrial and social applications (Glass and Vessey, 1995).

In taxonomy induction, the required domain knowledge can be acquired with many different methods for hypernym extraction, ranging from simple lexical patterns (Hearst, 1992; Oakes, 2005; Kozareva and Hovy, 2010) to statistical and machine learning techniques (Caraballo, 1999; Agirre et al., 2000; Ritter et al., 2009; Velardi et al., 2013). Recent efforts, such as Microsoft’s Probase (Wu et al., 2012) or the WebIsaDB (Seitner et al., 2016) similarly focus on ‘local’ extraction of single hypernym relations, and do not address the problem of how to combine these single relations into a coherent taxonomy. When taxonomies are automatically acquired, their cleaning (also called “pruning”) becomes a mandatory step (Velardi et al., 2013).

The contributions of this paper are two-fold:

1. We introduce a new algorithm, named ContrastMedium, which, given a noisy knowledge graph and its (possibly automatically generated) links to a companion taxonomy, is able to output a full-fledged taxonomy. Information from the reference taxonomy is projected onto the input noisy graph to automatically acquire topological clues, which are then used to drive the cleaning process. The reference taxonomy provides us with ground-truth taxonomic relations that make our knowledge-based method not truly unsupervised *sensu stricto*. However, the availability of resources like, for instance, WordNet (Fellbaum, 1998) or BabelNet (Navigli and Ponzetto, 2012) implies that these requirements are trivially satisfied;
2. We combine our approach with an unsupervised framework for knowledge acquisition from text (Faralli et al., 2016) to provide a full end-to-end pipeline for taxonomy induction from scratch.

2 Related Work

Knowledge Bases (KBs) can be created in many different ways depending on the availability of external resources and specific application needs. Recently, much work in Natural Language Processing focused on Knowledge Base Completion (Nickel et al., 2016a, KBC), the task of enriching and refining existing KBs. Many different methods have been explored for KBC, including exploitation of resources such as text corpora (Snow et al., 2006; Mintz et al., 2009; Aprosio et al., 2013) or other KBs (Wang et al., 2012; Bryl and Bizer, 2014) for acquiring additional knowledge. Alternative approaches, in contrast, primarily rely on existing information from the KB itself (Socher et al., 2013; Nickel et al., 2016b) used as ground-truth to simultaneously learn continuous representations of KB concepts and relations, which are used to infer additional KB relations. Finally, Open Information Extraction methods looked at ways to extract large amounts of facts from Web-scale corpora in order to acquire open-domain KBs (Etzioni et al., 2011; Faruqui and Kumar, 2015, *inter alia*);

In this paper, we focus on a different, yet complementary task, which is a necessary step when inducing novel KBs from scratch, namely extracting clean taxonomies from noisy knowl-

edge graphs. State-of-the-art algorithms differ by the amount of human supervision required and their ability to respect some topological properties while pruning. Approaches like those of Kozareva and Hovy (2010), Velardi et al. (2013) and Kapanipathi et al. (2014), for instance, apply different topological pruning strategies that require to specify the root and leaf concept nodes of the KB in advance – i.e., a predefined set of abstract top-level concepts and lower terminological nodes, respectively. The approach of Faralli et al. (2015) avoids such supervision on the basis of an iterative method that uses an efficient variant of topological sorting (Tarjan, 1972) for cycle pruning. Such lack of supervision, however, comes at the cost of not being able to preserve the original connectivity between the top (abstract) and the bottom (instance) concepts. Random edge removal (Faralli et al., 2015), in fact, can lead to disconnected components, a problem shared with the OntoLearn Reloaded approach (Velardi et al., 2013), which cannot ensure such property when used to approximate the solution for a large noisy graph.

Our work goes one step beyond the previous contributions by presenting a new efficient algorithm that is able to extract a clean taxonomy from a noisy knowledge graph without needing to know in advance – that is, having to manually specify – the top-level and leaf concepts of the taxonomy, while preserving the overall connectivity of the graph. We achieve this by projecting the information from a reference KB such as, for instance, WordNet (Fellbaum, 1998), onto the input noisy KB on the basis of pre-existing KB links – which in turn can be automatically generated with high precision using any of the existing solutions for KB mapping (Navigli and Ponzetto, 2012; Faralli et al., 2016, *inter alia*) or by relying on ground truth information from the Linguistic Linked Open Data cloud (Chiarcos et al., 2012).

Some aspects of the proposed approach – namely, the propagation of the nodes’ weights through the graph, which we metaphorically represent as the flow of a contrast medium across nodes (Section 3.3) – are somewhat similar in spirit to spreading activation (Collins and Loftus, 1975) and random walks on graphs (Lovász, 1993) approaches. However, in contrast to spreading activation approaches we leverage the graph directionality in order to reach all the possible nodes within the same connected components. More-

over, in contrast to random walks on graphs our method is deterministic in nature. Here, we argue for the choice of a deterministic approach, like ours, that does not require tuning of parameters: its termination is guaranteed by the number of iterations, which we bind by the maximal diameter $|E|$ for a graph $G = (V, E)$. Generally, random walk algorithms would provide an approximation that may lead to a less precise estimation of the order induced by the contrast medium level.

3 The ContrastMedium Algorithm

3.1 Problem Statement

Our work builds upon the notion of a **noisy knowledge graph** (NKG), which consists of a directed graph $G = (V, E)$ where V is a set of concepts and E the set of labelled binary semantic relations – e.g., those found between synsets like, for instance, hypernymy or meronymy within a semantic network like WordNet. In a NKG we assume both V and E to have been acquired automatically, e.g., in order to induce a domain-aware or a general purpose knowledge base. Additionally, we consider for our purposes the **hypernymy graph** $T = (T_V, T_E)$ of G , the subgraph made up of the hypernymy (i.e., *isa*-labeled) edges of E . Since T is a subgraph of G , we can expect that the former inherits a certain amount of noise from the latter.

Noise within hypernymy graphs can be further classified into: i) *noisy nodes*, the concepts that do not belong to a specific target vocabulary, e.g., domain concepts for domain-specific KBs, such as *Jaguar Cars* within a zoological taxonomy; ii) *noisy edges*, the wrongly-acquired relations between unrelated concepts or out-of-domain relations, e.g., *Jaguar Cars isa Feline*; iii) *cycles of hypernymy relations*, such as those derived from counts over very large corpora (Seitner et al., 2016), e.g., *jaguar (Panthera onca) → feline → animal → jaguar (Panthera onca)*. We accordingly define the task of extracting a clean taxonomy from a NKG as that of pruning the cycles, as well as the noisy edges and nodes, from the hypernymy subgraph T of G .

3.2 Resources Used

In order to enable end-to-end taxonomy induction from scratch, we combine our general approach with existing KBs that have been automatically induced from text and linked to reference lexical knowledge bases on the basis of unsuper-

vised methods. To this end, we use the linked disambiguated distributional KBs from Faralli et al. (2016)¹, which are built in three steps:

- 1) **Learning a JoBimText model.** Initially, a sense inventory is created from a large text collection using the pipeline of the JoBimText project (Biemann and Riedl, 2013).² The resulting structure contains disambiguated proto-concepts (i.e., senses), their similar and related terms, as well as aggregated contextual clues per proto-concept.
- 2) **Disambiguation of related terms.** Similar terms and hypernyms associated with a proto-concept are fully disambiguated based on the partial disambiguation from step (1). The result is a proto-conceptualization (PCZ), where all terms have a sense identifier.
- 3) **Linking to a lexical resource.** The PCZ is automatically aligned with an existing lexical resource (LR) such as WordNet or BabelNet. For example, `bridge:NN:3` is linked to the Babel synset `bn:00013077n` (the ‘infrastructure’ sense). That is, a mapping between the two sense inventories is created to combine them into a new extended sense inventory, a *hybrid aligned resource*.

Table 1 shows the proto-conceptualization entries for the polysemous terms *bridge* and *link*, namely their figurative (“`bridge:NN:2`” and “`link:NN:1`”) and concrete ‘infrastructure’ (“`bridge:NN:3`” and “`link:NN:0`”) senses, respectively. JoBimText models provide sense distinctions that are only partially disambiguated: the list of similar and hypernyms terms of each sense, in fact, does not carry sense information. Consequently, a semantic closure procedure is applied in order to obtain a PCZ and arrive at sense representation in which all terms get assigned a unique, best-fitting sense identifier (see Faralli et al. (2016) for details).

PCZs consist of a rich, yet noisy, disambiguated semantic network automatically induced from large amounts of text: links to existing lexical resources provide us a source of external supervision that can be leveraged to clean them and turn them into full-fledged taxonomies. Steps 1–3 are unsupervised by nature. Consequently, when

¹<https://madata.bib.uni-mannheim.de/171/>

²<http://www.jobimtext.org>

entry	similar terms	hypernyms
bridge:NN:2	gap:NN:2, divide:NN:2, link:NN:1, ...	issue:NN:2, topic:NN:3, ...
bridge:NN:3	road:NN:0, highway:NN:1, overpass:NN:3 ...	infrastructure:NN:1, project:NN:1, ...
link:NN:0	connection:NN:3, correlation:NN:1, linkage:NN:1 ...	service:NN:6, feature:NN:0, ...
link:NN:1	relationship:NN:1, interaction:NN:1, divide:NN:0 ...	problem:NN:1, topic:NN:3 ...

Table 1: Excerpt of a proto-conceptualization (PCZ) for the words “bridge:NN” and “link:NN”.

combined with our algorithm they provide a complete framework for fully unsupervised taxonomy induction from scratch. Note, however, that our approach offers a general solution to the problem of taxonomy cleaning. In an additional set of experiments, we apply it to different automatically generated taxonomies from a SemEval task in a more controlled setting where we rely on a few manually created KB links only.

3.3 The ContrastMedium Algorithm

At its core, our algorithm relies on the notion of a **linked noisy knowledge graph** (LNKG). This consists of a quintuple $(G, KB, KB_{root}, \lambda, M)$ where: i) $G = (V_G, E_G)$ is a noisy knowledge graph; ii) $KB = (V_{KB}, E_{KB})$ is a companion knowledge base providing a ground-truth taxonomy; iii) KB_{root} is the root node of the reference knowledge base KB (if several top-level nodes exist, an artificial root can be created by connecting them all); iv) λ is a conventional symbol to represent the “undefined concept”, i.e., a place-holder for empty mappings; v) $M : V_G \rightarrow V_{KB} \cup \{\lambda\}$ is the function, which maps nodes of V_G into nodes of V_{KB} or into the undefined concept λ . The key ideas behind ContrastMedium are:

- Identification of important topological clues from the companion knowledge base KB in order to hierarchically sort the concepts in G . For our purposes, KB is expected to be able to provide ground-truth taxonomic relations that can be safely projected onto G to guide the cleaning process: that is, we assume it to be reasonably clean. In contrast, we do not make any assumption on how KB has been created: our approach can be used with either manually created taxonomies like WordNet or (semi-)automatically induced ones, provided they are of sufficient quality. Hence, our method is knowledge-based without the need of further supervision other than that contained in KB ;
- Projection of topological clues from KB back onto the LNKG G on the basis of the links

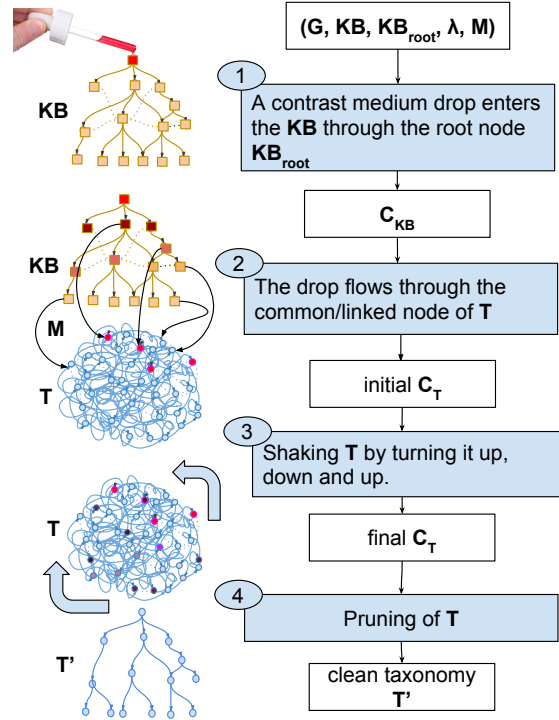


Figure 1: ContrastMedium: algorithm workflow.

found in the mapping M . Similarly to the case of the reference knowledge base, we do not make any assumption on how the links between G and KB have been created: while there exists different methods to automatically link (lexical) knowledge bases (Navigli and Ponzetto, 2012; Faralli et al., 2016), we later show that it is also possible to achieve state-of-the-art performance with a few manually given links;

- Propagation of the topological clues across the entire NKG G . That is, to cope with the partial coverage of automatic mappings, as well as the need to reduce the number of manually created KB links, we apply a signal propagation technique that solely relies on the structure of G ;
- To make use of the resulting topological clues to drive the taxonomy pruning process. That is, propagated topological clues from KB are additionally leveraged to ensure that the output

ALGORITHM 1: The ContrastMedium algorithm.

Input: $(G = (V, E), KB = (V_{KB}, E_{KB}), KB_{root}, \lambda, M)$
Output: hypernymy graph T' of G , s.t. T' has no cycles.
// Estimating clues from KB (Fig. 1, step 1)
1 $\forall x \in V_{KB}: C_{KB}(x) = 0;$
2 injectContrastMedium(KB, KB_{root});
// Transferring clues from KB to G (Fig. 1, step 2)
3 $T = (V_T, E_T) \leftarrow$ hypernymyGraph(G);
4 $\forall x \in V_T: C_T(x) = 0;$
5 transferClues(M, KB, T, C_{KB}, C_T);
// Shaking the graph T (Fig. 1, step 3)
6 shake(UP, T, C_T); // propagate through in-edges
7 shake(DOWN, T, C_T); // propagate through out-edges
8 shake(UP, T, C_T); // propagate through in-edges
// Pruning the graph T (Fig. 1, step 4)
9 $T' =$ prune(T, C_T);
10 return T' ;

results in a proper taxonomic structure.

We rely on the metaphor of a contrast medium (CM) to describe our approach, which is summarized in Figure 1. In the context of clinical analysis, a CM is injected into the human body to highlight specific complex internal body structures (in general, the venous system). In a similar fashion, we detect the topological structure of a graph by propagating a certain amount of CM that we initially inject through the node KB_{root} of the companion knowledge base KB . The highlighted structure indicates the distance of a node with respect to the node KB_{root} . Then the lowest values of contrast medium indicate the leaf terminological nodes. The observed quantities are then transferred to corresponding nodes of the noisy graph by the mapping M . Next, the medium is propagated by ‘shaking’ the noisy graph. We let the fluid reach all the components G by alternating two phases of propagation: letting the CM to flow through both incoming (‘shake up’); and outgoing (‘shake down’) edges. At the end, we use the partial order induced by the observed node level of CM to drive the pruning phase, and ‘stretch’ the original NKG G into a proper DAG.

Our approach is presented in Algorithms 1 and 2.³ It consists of the following main steps:

1) CM injection Cf. Figure 1, block 1 and Algorithm 1, lines 1-2. We initially define the function $C_{KB} : V_{KB} \rightarrow [0.0 - 1.0]$ and assign a zero contrast medium level to all the nodes of the KB graph $C_{KB}(x) = 0, x \in V_{KB}$ (line 1). Next,

³A demo is available at <http://web.informatik.uni-mannheim.de/faralli/cm.html> with examples of the application of ContrastMedium to a few simple LNKGs.

ALGORITHM 2: The Shake routine.

Input: *direction* may be UP or DOWN,
 $graph = (V_{graph}, E_{graph}), C_{graph}$
Output: the updated C_{graph}
1 **foreach** $x \in V_{graph}$ **do**
2 | $Current_{graph}(x) = C_{graph}(x); Flown_{graph} = 0.0;$
// iteratively propagates the CM
3 **for** $i = 0$ to $|E_{graph}| - 1$ **do**
4 | **foreach** $x \in V_{graph}$ **do**
5 | | $InOut_{graph}(x) = 0.0;$
6 | **foreach** x s.t. $Current_{graph}(x) > 0.0$ **do**
7 | | $CMlevel = Current_{graph}(x);$
8 | | **if** $direction == DOWN$ **then**
9 | | | $O = outgoingEdges(x, graph);$
10 | | | **foreach** $(x, y) \in O$ **do**
11 | | | | $InOut_{graph}(y) += \frac{CMlevel}{max(|O|, 1)};$
12 | | | **else**
13 | | | | $I = incomingEdges(x, graph);$
14 | | | | **foreach** $(y, x) \in I$ **do**
15 | | | | | $InOut_{graph}(y) += \frac{CMlevel}{max(|I|, 1)};$
16 | | | $Flown_{graph}(x) += CMlevel;$
17 | **foreach** $x \in V_{graph}$ **do**
18 | | $Current_{graph}(x) = InOut_{graph}(x);$
19 **foreach** $x \in V_{graph}$ **do**
20 | $C_{graph}(x) = Flown_{graph}(x);$

we call the routine ‘injectContrastMedium’ which: 1) assigns an initial contrast level equals to 1.0 to the node KB_{root} of the KB graph; ii) uses the routine ‘Shake’ with the direction parameter equals to ‘DOWN’ (see Algorithm 2 and Step 3 ‘Graph shaking’ for more details) to let the CM drop through KB . In practice, the shaking routine implements a node contrast medium level propagation algorithm following the outgoing (‘down’) or the incoming (‘up’) edges of the graph.

2) CM transfer Cf. Figure 1, block 2 and Algorithm 1, lines 3-5. In the next phase, we first extract the hypernymy subgraph $T = (V_T, E_T)$ of G (see Section 3.1) and then follow the links in the mapping M to transfer the contrast medium levels, i.e., $C_T(y) = C_{KB}(x)$ (s.t. $x \in V_{KB}, y \in V_T, (y \rightarrow x) \in M$).

3) Graph shaking Cf. Figure 1, block 3 and Algorithm 1, lines 6-8. After having transferred the CM to the target hypernym graph T of G , we shake T to let the CM flow by traversing the incoming, the outgoing, and finally the incoming edges again – see Algorithm 2 for details on the ‘Shake’ routine. Note that these two kinds of propagation are needed since the CM needs to be propagated through all the nodes of the graph to highlight the topological clues we are searching for. In particular, in Algorithm 2 at each iteration t for each node $x \in V_{graph}$, depending on

the value of the parameter *direction* (line 8 and line 12): i) we observe a CM level for the node x (line 7); ii) if *direction* == DOWN (lines 9-11) we traverse all the outgoing edges (x, y) of x and propagate the observed CM level of x , otherwise (*direction* == UP, lines 13-15) we traverse the incoming edges (y, x) and propagate the CM level to the nodes y ; iii) the value of $Flown_{graph}(x)$ is incremented by the observed CM level (line 16); iv) for each node x we reset the current observed value of the CM level with the portion of the liquid which has flown from the incoming or the outgoing edges during the propagation (lines 17-18).

Depending on the propagation direction, we have two different behaviours for the CM. When exiting a node x through out the outgoing edges (*direction* == DOWN) we increment the level of contrast medium of the reached nodes by the observed value of x divided by number of outgoing edges of x . By converse, when we climb (*direction* == UP) across the incoming edges of a node x we increment the CM level of the reached node by the observed CM quantity of x divided by the number of incoming edges of x .

Note that the sequence UP/DOWN/UP and the specular DOWN/UP/DOWN are the only ones from the 8 possible combinations which can guarantee the contrast medium to flow on the entire graph. We simply selected the first sequence since the final rank places candidate root nodes on the top (and candidate leaf nodes on the bottom).

4) Pruning Cf. Figure 1, block 4 and Algorithm 1, lines 9. Finally, we create a clean taxonomy T' by pruning the graph T on the basis of the contrast levels found in C_T . CM levels in C_T can be used to induce a order of the nodes that, intuitively, captures the level of conceptual abstraction for the nodes in T . We use them to produce a clean taxonomy as follows. We first sort the nodes $v \in V_T$ in a list $S = s_0, s_1, \dots, s_{|V_T|-1}$ by the decreasing resulting CM level value in C_T . The nodes with a higher level of contrast medium are candidates to be at the top level while the ones at the end of the list are candidates to be leaf nodes of the output taxonomy. Next, the pruning routine starts from a graph $T' = (V_{T'} = V_T, E_{T'} = \emptyset)$ and for each node $s \in S$ (from the last node to the first) add to $E_{T'}$ all the edges of the kind $e = (y, s)$ where a path from y to s does not exists in T and with y belonging to one of the following: i) the set of peers $\{x \in S \text{ s.t. } C_T(x) = C_T(s)\}$; ii) the

ascending ordered list of preceding ($x \in S \text{ s.t. } C_T(x) > C_T(s)$); iii) the ascending ordered list of following ($x \in S \text{ s.t. } C_T(x) < C_T(s)$)

Complexity analysis. The propagation step (Figure 1, blocks 1 and 3; Algorithm 2) costs $O(|E| * |V|)$ since we iteratively analyze all the nodes of V for a number $|E|$ of iterations. The final step of pruning (Figure 1, block 4), instead, can have a time cost $O(|V|^2 * (|E| + |V|))$, since, in the worst case, the algorithm must analyse all the possible pairs of vertices, and then test the existence of a directed path between the candidate pairs of nodes.

4 Experiments

We perform two sets of experiments. We first evaluate our approach when applied to large, automatically induced noisy knowledge graphs (Section 4.1) and then quantify the impact it can have to further improve the quality of the output of state-of-the-art taxonomy induction systems (Section 4.2).

4.1 Experiment 1: Pruning existing LNKG

We first apply ContrastMedium to a variety of knowledge graphs that have been automatically acquired and linked to reference KBs like WordNet and BabelNet using unsupervised methods (Section 3.2). Our research questions (RQs) are:

RQ1 Can we use ContrastMedium as component of a complete framework for fully unsupervised taxonomy induction from scratch?

RQ2 What is the quality of the resulting taxonomies?

4.1.1 Experimental Setting

We apply our pruning algorithm to the automatically acquired KBs presented by Faralli et al. (2016). These noisy knowledge graphs have been induced from large text corpora and include both taxonomic and other (i.e., related, topically associative) semantic relations (cf. Table 1), as well as automatically induced mappings to lexical knowledge bases like WordNet and BabelNet. These NKGs have been induced from a 100 million sentence news corpus (*news*) and from a 35 million sentence Wikipedia corpus (*wiki*), using different parameter values to generate sense inventories of different granularities (e.g., 1.8 vs. 6.0 average senses per term for the wiki-p1.8 and wiki-p6.0 datasets, respectively). Table 2 shows some of

dataset	senses		polysemy		hypernyms		links	hypernymy graph	
	#	avg.	max	#	avg.	#	$ V_T $	$ E_T $	
news-p1.6	332k	1.6	18	15k	6.9	60k	170k	1.538k	
news-p2.3	461k	2.3	17	15k	5.8	95k	225k	1.871k	
wiki-p1.8	368k	1.8	15	15k	4.4	67k	185k	1.167k	
wiki-p6.0	1.5M	6.0	36	52k	1.7	279k	394k	1.901k	

Table 2: Dimensions of the four datasets adopted as linked noisy knowledge graphs (Faralli et al., 2016).

the dimensions for each of the four NKGs – number of senses, average and maximum sense polysemy, number and average hypernyms per sense, the number of linked senses to WordNet concepts (i.e., “links”), and the number of nodes and edges for the corresponding hypernymy graph. Since our algorithm primarily focuses on conceptual hierarchical (taxonomic) structures – referred to as the TBox in Knowledge Representation – we use the WordNet mappings only, since the manual inspection of the BabelNet mappings revealed that they are focused primarily on instances (that is, ABox statements). In order to have a complete quintuple for each NKG, we selected, for the companion KB, the top KB_{root} concept `entity` of the WordNet taxonomy (SynsetID SID-00001740-N).

4.1.2 Measures

We benchmark ContrastMedium using a variety of metrics that are meant to capture structural properties of the output taxonomies (to describe the impact of pruning on the original NKGs), as well as an estimation of their overall quality.

Edge compression: the ratio of the number of pruned edges over the total number of edges:

$$C_{E_G, G'} = \frac{|E_G| - |E_{G'}|}{|E_G|}$$

where E_G and $E_{G'}$ represent the number of edges found within the input (G) and pruned (G') taxonomy, respectively.

Pruning accuracy: the performance on a 3-way classification task to automatically detect the level of granularity of a concept as a proxy to quantify the overall quality of the output taxonomies. Pruning accuracy is estimated using gold-standard annotations that are created from a random sample of 1,000 nodes for each NKG. Two annotators with previous experience in knowledge acquisition and engineering were asked to provide for each

concept whether it can be classified as: i) a root, top-level abstract concept – i.e., any of `entity`, `object`, etc. and more in general nodes that correspond to abstract concepts that we can expect to be part of a core ontology such as, for instance, DOLCE (Gangemi et al., 2002); ii) a leaf terminological node (i.e., instances such as `Lady Gaga` or `Porsche 911`); iii) or a middle-level concept (e.g., `celebrity` or `cars`, concepts not fitting into any of the previous classes). An adjudication procedure was used to resolve any discrepancy between the two annotators: the inter-annotator agreement after adjudication is $\kappa = 0.657$ (Fleiss, 1971), with most disagreement occurring on the identification of abstract, core ontology concepts.

A local 3-way classification task provides a rather crude way to estimate the performance on inducing hierarchical structures like taxonomies. Here, we use it primarily to benchmark how well ContrastMedium compares against other, structure-agnostic algorithms used within state-of-the-art solutions such as, for instance, Tarjan’s topological sorting (Section 2), which only break cycles in a random fashion.

Given ground-truth concept granularity judgements, we compute standard accuracy for each of the three classes. That is, we compare the system outputs against the gold standards and obtain three accuracy measures: one for the root nodes (A_R), one for the nodes ‘in the middle’ (A_M) and finally one for the leaf nodes (A_L). For example a true positive root node is a node annotated as a root node in the gold standard and having no incoming edges in the pruned graph.

Error Reduction (ER): finally, we compute the relative error reduction of ContrastMedium against other, baseline approaches as:

$$\frac{Baseline_{errors}/|sample| - CM_{errors}/|sample|}{Baseline_{errors}/|sample|}$$

As *baseline* we use the approach of Faralli et al.

dataset	Pruned Knowledge Graph						Pruning accuracy						ER
	ContrastMedium			Tarjan (baseline)			ContrastMedium			Tarjan (baseline)			
	$ V_{G'} $	$ E_{G'} $	$C_{E_{G'},G'}$	$ V_{G'} $	$ E_{G'} $	$C_{E_{G'},G'}$	A_R	A_M	A_L	A_R	A_M	A_L	
news-p1.6	170k	1.536k	0.15%	170k	1.535k	0.18%	98.9	98.3	99.3	93.3	94.6	95.3	0.62
news-p2.3	225k	1.867k	0.19%	225k	1.866k	0.23%	98.7	98.7	99.9	95.7	94.7	95.6	0.50
wiki-p1.8	183k	1.165k	0.18%	183k	1.164k	0.22%	97.6	94.7	97.3	93.1	87.3	94.1	0.41
wiki-p6.0	394k	1.897k	0.18%	394k	1.896k	0.21%	95.9	94.3	98.3	89.5	90.1	92.8	0.50

Table 3: Structural analysis, pruning accuracies and error reduction (ER) for the four LNKGs.

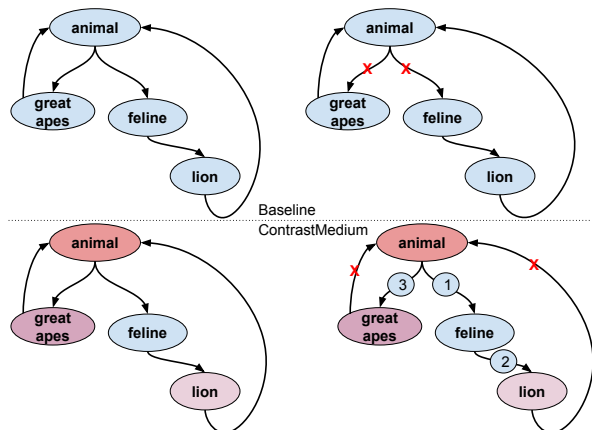


Figure 2: An example noisy graph and the different solutions provided by ContrastMedium and the baseline.

(2015) based on Tarjan’s topological sorting (Section 2), which iteratively searches for a cycle (until no cycle can be found) and randomly removes an edge from it. To the best of our knowledge, this is the only algorithm that we can fairly compare with, since alternative solutions all need to know the sets of root and leaf nodes in advance.

4.1.3 Results and discussion

Table 3 summarizes the performance of ContrastMedium on the four automatically acquired NKGs. The results show that the pruning impact of our approach is lower than that of the baseline (an average of 1K edges of difference, cf. columns 3 and 6), which also determines higher edge compression $C_{E_{G'},G'}$ values for the baseline method. Despite being less aggressive in terms of the number of edges pruned, ContrastMedium outperforms the Tarjan-based algorithm on all datasets in terms of accuracy. Thanks to our method, in fact, we are able to achieve, even despite the baseline already reaching very high performance levels (well above 90% accuracy), improvements of up to 6 points, with an overall error reduction between around 40% and 60%. To provide an

intuition of why ContrastMedium clearly outperforms the baseline approach, we provide in Figure 2 an exemplified depiction of a typical case on which the baseline fails (based on a manually inspected random sample). In our example, the Tarjan baseline first detects the cycle $C_1 = (\text{lion} \rightarrow \text{animal} \rightarrow \text{feline} \rightarrow \text{lion})$ and randomly decides to break it by removing the edge $(\text{animal} \rightarrow \text{feline})$. Next, it detects the cycle $C_2 = (\text{animal} \rightarrow \text{great apes} \rightarrow \text{animal})$ and randomly decides to break it by removing the edge $(\text{animal} \rightarrow \text{great apes})$. ContrastMedium, instead, after the shaking of the graph can leverage the partial ordering of the nodes (based on the concept granularity of the corresponding concepts) to select the edges $(\text{animal}, \text{feline})$, $(\text{feline}, \text{lion})$ and $(\text{animal}, \text{great apes})$, while removing all remaining wrong and redundant edges.

4.2 Experiment 2: SemEval-15 task 17

We next evaluate the overall impact of our approach within an existing benchmark for the taxonomy induction task. Intuitively, most of the benefits from our method derive from the “gold standard” information of the companion KB, and its linking to the NKG, which act as a source of supervision. Consequently, we address the research question of how much (pseudo-)supervision our method needs in terms of KB links, and whether it can be used to improve the state-of-the-art on the task of taxonomy induction.

4.2.1 Experimental Setting

We use the benchmark data from the SemEval-15 task 17 “Taxonomy Extraction Evaluation: TExEval” (Bordea et al., 2015), since it provides us with gold-standard datasets and system outputs within a standard, easy-to-reproduce setting. Initially, we select from the participating systems⁴ the two best performing taxonomies based on the Cumula-

⁴Cf. Table “Comparative Evaluation” at <http://alt.qcri.org/semeval2015/task17/index.php?id=evaluation>

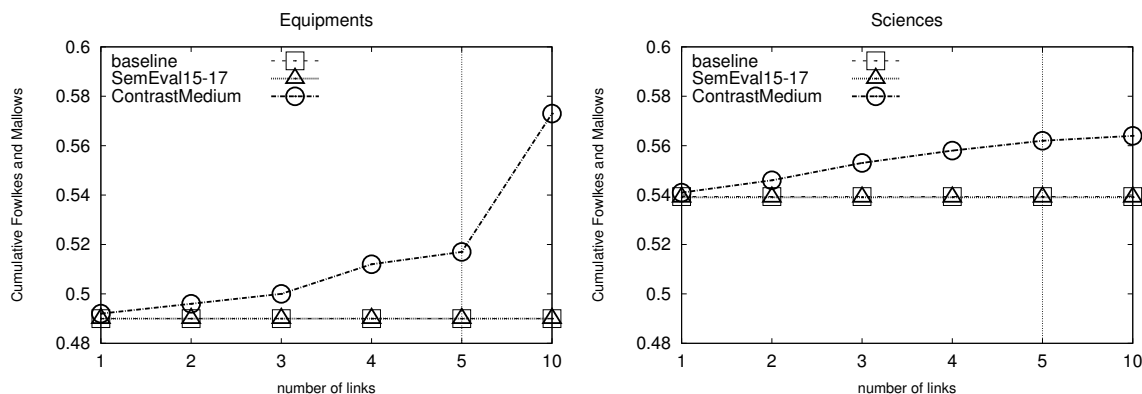


Figure 3: Performance on the SemEval-15 TExEval dataset (Cumulative Fowlkes&Mallows measure).

tive Fowlkes&Mallows (CF&M) measure (Velardi et al., 2012), the *Equipments* and *Sciences* taxonomies from the INRIASAC and the LT3 teams respectively. We next apply our approach to these taxonomies, in order to clean them in a post-processing fashion. By selecting the top-systems we can see how far we can advance the state-of-the-art overall. Besides, these two taxonomies are also the ones containing the highest number of cycles, giving the application of our cleaning algorithm a more challenging (and meaningful) setting. To remove the effects of automatic linking and quantify the amount of manual efforts needed by our approach, 10 random concepts from each of these resources are manually linked to WordNet, and the taxonomies subsequently pruned using ContrastMedium and the baseline. We then evaluate performance following the task’s experimental setting and compute the CF&M measure for different levels of manually-created KB links.

4.2.2 Results and discussion

In Table 3, we report the performance on the SemEval task for the two selected input taxonomies. Results on the structural similarities of the pruned taxonomies with the gold standard ones, computed using the CF&M measure, indicate that, thanks to ContrastMedium and with a minimal human effort – the creation of just a few KB links (up to 10), which are needed only when automatic linking is not available – it is possible to boost the quality of taxonomies using state-of-art methods by a large margin. For instance, in the case of the *Equipments* taxonomy, we improve up to 7 points. The baseline, which only breaks cycles, is not able to reassess the graph structure and only provides very small improvements to the submitted NKGs.

Overall, the results show that ContrastMedium leads to competitive performance on a hard, realistic benchmark such as TExEval, achieving the best overall results for both taxonomies. That is, our algorithm is able to improve the state-of-the-art on taxonomy induction by additionally boosting the quality of existing top-performing systems for this task: this is achieved on the basis of a minimally supervised approach that only requires a few links to a reference KB, which is used to provide ground-truth taxonomic relations and guide the cleaning process.

5 Conclusions

In this paper, we presented ContrastMedium, a novel algorithm that can be applied to automatically linked noisy knowledge graphs to provide an end-to-end solution for fully unsupervised taxonomy induction from scratch, i.e., without any human effort. Our results indicate that ContrastMedium can be successfully applied to a wide range of automatically acquired KBs, ranging from large linked noisy knowledge graphs all the way to small-scale induced taxonomies to produce high-quality *isa* hierarchies that achieve state-of-the-art results on SemEval benchmarks. As future work, we plan to improve the scalability of the algorithm, in particular its time complexity order, and apply it to Web-scale resources like the WebIsaDB (Seitner et al., 2016) or state-of-the-art approaches like TAXI (Panchenko et al., 2016), as well as to publicly release the created resources.

Acknowledgments

We acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) under the JOINT project.

References

- Eneko Agirre, Olatz Ansa, Eduard H. Hovy, and David Martínez. 2000. Enriching very large ontologies using the WWW. In *Proceedings of the ECAI 2000 Workshop on Ontology Learning*.
- Alessio P. Arosio, Claudio Giuliano, and Alberto Lavelli. 2013. Extending the coverage of DBpedia properties using distant supervision over Wikipedia. In *Proceedings of the NLP & DBpedia workshop co-located with the 12th International Semantic Web Conference (ISWC 2013)*, pages 20–31.
- Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1):55–95.
- Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia – A crystallization point for the web of data. *Journal Web Semantics*, 7(3):154–165.
- Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.
- Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (TExEval). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 902–910.
- Volha Bryl and Christian Bizer. 2014. Learning conflict resolution strategies for cross-language wikipedia data fusion. In *Proceedings of the 23rd International World Wide Web Conference*, pages 1129–1134.
- Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126.
- Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1306–1313.
- Christian Chiarcos, Sebastian Hellmann, and Sebastian Nordhoff. 2012. Linking linguistic resources: Examples from the Open Linguistics Working Group. In Christian Chiarcos, Sebastian Nordhoff, and Sebastian Hellmann, editors, *Linked Data in Linguistics - Representing and Connecting Language Data and Language Metadata*, pages 201–216. Springer.
- Allan M. Collins and Elizabeth F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review*, 82(6):407 – 428.
- Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.
- Arnab Dutta, Christian Meilicke, and Simone Paolo Ponzetto. 2014. A probabilistic approach for integrating heterogeneous knowledge sources. In *Proceedings of the 11th Extended Semantic Web Conference*, pages 286–301.
- Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 3–10.
- Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545.
- Stefano Faralli and Roberto Navigli. 2013. Growing Multi-Domain Glossaries from a Few Seeds using Probabilistic Topic Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 170–181.
- Stefano Faralli, Giovanni Stilo, and Paola Velardi. 2015. Large scale homophily analysis in twitter using a twixonomy. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 2334–2340.
- Stefano Faralli, Alexander Panchenko, Chris Biemann, and Simone P. Ponzetto. 2016. Linked disambiguated distributional semantic networks. In *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference*, pages 56–64.
- Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1351–1356.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.
- Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.
- Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. 2002. Sweetening ontologies with DOLCE. In *Proceedings of the 13th International Conference on*

- Knowledge Engineering and Knowledge Management*, pages 166–181.
- Robert L. Glass and Iris Vessey. 1995. Contemporary application-domain taxonomies. *IEEE Software*, 12(4):63–76.
- Rahul Gupta, Alon Y. Halevy, Xuezhong Wang, Steven Euijong Whang, and Fei Wu. 2014. Biperpedia: An ontology for search applications. In *Proceedings of the 40th International Conference on Very Large Data Bases*, pages 505–516.
- Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, pages 539–545.
- Johannes Hoffart, Fabian Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194(28):28–61.
- Pavan Kapanipathi, Prateek Jain, Chitra Venkataramani, and Amit Sheth. 2014. User interests identification on Twitter using a hierarchical knowledge base. In *The Semantic Web: Trends and Challenges*, pages 99–113. Springer.
- Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118.
- László Lovász. 1993. Random walks on graphs: A survey. *Combinatorics*, 2:146.
- Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.
- Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.
- Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.
- Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1955–1961.
- Michael P. Oakes. 2005. Using Hearst’s rules for the automatic acquisition of hyponyms for mining a pharmaceutical corpus. In *Proceedings of the RANLP 2005 Text Mining Workshop*, pages 63–67.
- Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cédric Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: A taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, pages 1320–1327.
- Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the AAAI Spring Symposium on Learning by Reading and Learning to Read*, pages 88–93.
- Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Paolo Ponzetto. 2016. A large database of hypernymy relations extracted from the web. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 360–367.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808.
- Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.
- Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160.
- Paola Velardi, Roberto Navigli, Stefano Faralli, and Juana María Ruiz-Martínez. 2012. A new method for evaluating automatically learned terminological taxonomies. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 1498–1504.
- Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction. *Computational Linguistics*, 39(3):665–707.
- Zhichun Wang, Juanzi Li, Zhigang Wang, and Jie Tang. 2012. Cross-lingual knowledge linking across wiki knowledge bases. In *Proceedings of the 21st International World Wide Web Conference*, pages 459–468.
- Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 481–492.

Probabilistic Inference for Cold Start Knowledge Base Population with Prior World Knowledge

Bonan Min and Marjorie Freedman and Talya Meltzer *

Raytheon BBN Technologies

10 Moulton St, Cambridge, MA 02138, USA

{bonan.min, marjorie.freedman}@raytheon.com

Abstract

Building knowledge bases (KB) automatically from text corpora is crucial for many applications such as question answering and web search. The problem is very challenging and has been divided into sub-problems such as mention and named entity recognition, entity linking and relation extraction. However, combining these components has shown to be under-constrained and often produces KBs with supersize entities and common-sense errors in relations (a person has multiple birthdates). The errors are difficult to resolve solely with IE tools but become obvious with world knowledge at the corpus level. By analyzing Freebase and a large text collection, we found that per-relation cardinality and the popularity of entities follow the power-law distribution favoring flat long tails with low-frequency instances. We present a probabilistic joint inference algorithm to incorporate this world knowledge during KB construction. Our approach yields state-of-the-art performance on the TAC Cold Start task, and 42% and 19.4% relative improvements in F1 over our baseline on Cold Start hop-1 and all-hop queries respectively.

The third author is currently with The Affinity project. This work was done while she was at BBN.

This work was supported by DARPA/I2O Contract No. FA8750-13-C-0008 under the DEFT program. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

1 Introduction

Automatically transforming a large corpus into a structured knowledge base (KB) has long been a goal of information extraction (IE) research. KB population incorporates many IE tasks including named entity recognition, entity linking and relation extraction, each of which rely on deeper linguistic analysis, e.g., syntactic parsing or anaphora resolution. Since 2012, NIST ¹ has run an open shared task in KB population (KBP) under TAC ². Most participating systems (Mayfield et al., 2014; Min et al., 2015; Roth et al., 2015; Angeli et al., 2014; Nguyen et al., 2014; Monahan and Carpenter, 2012) combine many independent components to perform the full task.

As will be familiar to most IE researchers, the individual components are not perfect. When combined into a pipeline, errors compound. We found that a KB produced with a simple combination of state-of-the-art IE components (Ramshaw et al., 2011) is very sensitive to component-level errors (Grishman, 2013).

Table 1 illustrates a real entity coreference mistake. *Barack Obama* and *Ehud Barak* were incorrectly linked because of ambiguous context and high lexical overlap. The mistake leads to erroneous facts about employment, familial relations, etc. We see additional mistakes when we review the names in those entities with the most mentions: the *U.S.* entity contains more than 20,000 mentions. 85% are correct (e.g., *United States, U.S.*), but there is a long tail of incorrect yet infrequent (each accounting for < 1%) mentions linked to the entity e.g., *North America, Latin American*. We also see counter-intuitive errors in relation extraction: 5% of person entities have multiple birthdates; the KB asserts 8 spouses for an infrequently

¹U.S. National Institute of Standards and Technology

²Text Analysis Conference: www.nist.gov/tac/

mentioned entity. Similar errors have been reported in (McNamee et al., 2013) and (Singh et al., 2013b).

Named mentions of PER: <i>Barack Obama</i> : <i>Barack, Barack Obama, Ehud Barak, Barak</i>
Text: <i>Barak endorses Barack, ... Defense Minister Ehud Barak said Barack Obama has been the most supportive president on Israeli security</i>

Table 1: Example of entity linking errors.

Analyzing these errors suggests a limitation of performing KB population solely with IE tools. These mistakes only become obvious in the context of external world knowledge with the full set of facts extracted from many documents, e.g. when applying our expectations about the cardinality of a relation. With just a single document, resolving these mistakes requires challenging inference (Ji et al., 2005).

In this paper, we present a probabilistic framework to incorporate real-world knowledge into Cold Start KB population. Our contributions include:

- Identifying from real world datasets that entity popularity and each relation’s cardinality follow the power-law distribution with long tails of low-frequency instances.
- Defining a corpus-level joint objective for KBP that incorporates multiple IE components and prior world knowledge on entity popularity and per-relation cardinalities, and showing the prior knowledge helps to reduce errors.
- Outperforming the top-ranked entry in Cold Start 2015.

The paper is organized as follows: we first introduce the Cold Start KBP task, then present the joint probabilistic framework, followed by analysis of the world knowledge and how to incorporate it. We then describe our inference algorithm. Lastly, we present experimental results, related work, and conclude with suggestions for future research.

2 Cold Start KBP

The schema consists of 3 entity types (person, organization, and GPE) and 42 slots (relation classes)³. Systems start with an empty KB (cold start) and populate it according to the schema with

³We will use *slot* and *relation* interchangeably in this paper.

information extracted from the corpus. All facts in the KB must be grounded with justifying text from the corpus.

A KB entity is defined as a cluster of mentions that refer to the same real-world entity, e.g., *Smith*, *John Smith*, and *John H Smith* are 3 mentions for the entity *John_H_Smith*. Every named mention of an entity is recorded. A relation is a triplet (*subject*, *slot*, *object*), where *subject* and *object* are entities,⁴ and *slot* is the relation between them. For example, (*Bart Simpson*, *per:siblings*, *Lisa Simpson*) is the relation *Bart Simpson is a sibling of Lisa Simpson*. A relation’s provenance points to up to 4 snippets in the corpus that justify the relation. The evaluation process is described in the Experiments Section.

3 A Probabilistic Framework

Following most Cold Start KBP systems (Mayfield et al., 2014; Min et al., 2015; Roth et al., 2015; Angeli et al., 2014; Nguyen et al., 2014; Monahan and Carpenter, 2012), our baseline uses a cascade of NLP components from document-level analysis to corpus-level aggregation. We run BBN’s SERIF (Ramshaw et al., 2011) for mention, value and name tagging, coreference resolution, sentence-level relation extraction, alongside other analysis such as syntactic parses. Then we aggregate entities with entity discovery and linking and relations with relation extraction.

Given a set of pre-trained NLP components, the process is essentially an inference task. We introduce the following notation:

- M is the list of mentions
- E is the set of entities to populate the KB
- R is the set of relation types.
- x_i is the observed text for mention i , $x_i \in M$
- u_i is entity ID from the KB assigned to mention i , $i \in \{1, 2, \dots, |M|\}$, $u_i \in \{1, 2, \dots, |E|\}$
- $z_{i,j} = r$ indicates the relation r between a pair of mentions $x_i, x_j \in M$ and $r \in R \cup \{Other\}$
- $y_{i,j}^r$ is an indicator variable: $y_{i,j}^r = 1$ if a relation $r \in R$ exist between entity pair $\langle e_i, e_j \rangle$, $i, j \in \{1, 2, \dots, |E|\}$, and 0 otherwise.

The key steps in the pipeline are the following:

Mention extraction: We use a structured perceptron model (Ramshaw et al., 2011) to extract named mentions M .

⁴A small number of the relations take values not entities. We do not differentiate in this work.

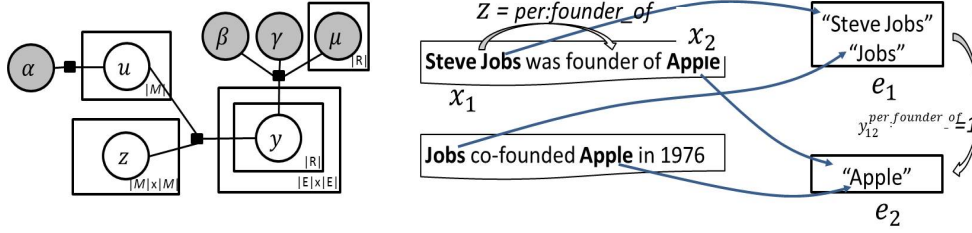


Figure 1: A simplified plate model of the probabilistic model(left), and an example (right) illustrates the KB construction process with aligned variables. The plate model only shows RE and EDL factors, and factors incorporating world knowledge. The example(right) compensates by showing the process without priors.

Entity Discovery & Linking (EDL): This step creates a candidate entity set E for the KB and infers which document entity (represented by its named mentions) is associated with which KB entity, i.e. assigning values for $\{u_i\}$. We use a sieve-like (Raghunathan et al., 2010) algorithm for in-document coreference. For simplicity, we only model EDL of names⁵.

We define potential functions over variables $\{u_i\}$ for each pair of mention x_i and the j th entity e_j :

$$\Psi_i^{EDL}(u_i = j|x_i) = \exp(\sum_k \theta_k \phi_k(u_i = j|x_i))$$

The baseline system solves the EDL problem by inferring $u_i^* = \arg \max \Psi_i^{EDL}(u_i)$. It uses a name database collected from Freebase (Bollacker et al., 2008) and GeoNames⁶. First, it clusters novel names to create new candidate entities in addition to entries in the name database. A novel name is defined as a name that could not be resolved to a database entry. It then rescans the corpus and links each document-level entity to a corpus-level entity (an entry in the name database or a novel name). The EDL model Ψ^{EDL} uses features such as string edit distance and indicators representing whether appearing in the same name variant set. $\{\theta_k\}$ and $\{\phi_k\}$ are the weights and feature functions respectively.

Mention-level Relation Extraction (MRE): This step infers which relation $z_{i,j} = r (r \in R \cup \{Other\})$ exists between each pair of mentions $\langle x_i, x_j \rangle$, we define potential functions:

$$\Psi_{i,j}^{MRE}(z_{i,j} = r|x_i, x_j) = \exp(\sum_k \theta'_k \phi'_k(z_{i,j} = r|x_i, x_j))$$

We run several relation finding algorithms (Min et al., 2015), including statistical models trained

⁵Decisions made for named mentions will be applied to the corresponding document-level entities.

⁶www.geonames.org

from ACE⁷ relation annotation and distant supervision (Mintz et al., 2009) in which we align Freebase pairs into Gigaword (Parker et al., 2011) to generate training examples, and a pattern matcher with a few hand-written patterns that capture local contexts.

To train a log-linear model Ψ^{MRE} for combining these algorithms and to tune the confidences of their extractions, we follow (Viswanathan et al., 2015) and train a stacked classifier using output and confidences of the extractors. We use assessment datasets from TAC Cold Start KBP 2013 and 2014, and Slot Filling evaluations in 2013 and 2014. The features we used are: source algorithm name, slot, confidence score (if exists), argument mention level (*pronoun, name, or nominal*), lexical sequence between pair of arguments, propositional path between the pair of arguments. $\{\theta'_k\}$ and $\{\phi'_k\}$ are the weights and feature functions respectively.

Relation Extraction (RE): This aggregation step infers which relations exist between each pair of entities at the KB level, i.e. assigning values for $\{y_{i,j}^r\}$. We define the potential functions over the indicator variables $\{y_{i,j}^r\}$, by looking at all pairs of mentions $x_m, x_n \in M$, their potential to have a relation r and likelihood to be associated with entities $e_i, e_j \in E$:

$$\begin{aligned} \Psi_{i,j,r}^{RE}(y_{i,j}^r = 1|x) = \\ \max_{\langle m,n \rangle} (\Psi_m^{EDL}(u_m = i|x_m) \Psi_n^{EDL}(u_n = j|x_n) \\ \Psi_{m,n}^{MRE}(z_{m,n} = r|x_m, x_n)) \end{aligned}$$

and $\Psi_{i,j,r}^{RE}(y_{i,j}^r = 0|x) = \Psi_0^{RE}$ where Ψ_0^{RE} is a parameter learned from previously seen data. The aggregation from a set of $z_{m,n}$ to a set of $y_{i,j}$ is similar to noisy-or relation aggregation (Hoffmann et al., 2011; Riedel et al., 2010; Sur-

⁷itl.nist.gov/iad/mig/tests/ace/2005/

deanu et al., 2012) and supports overlapping relations (Hoffmann et al., 2011; Surdeanu et al., 2012).

The joint distribution defined over the full set of variables u, y is:

$$Pr(u, y|x) \propto \prod_m \Psi_m^{EDL}(u_m|x_m) \cdot \prod_{i,j,r} \Psi_{i,j,r}^{RE}(y_{ij}^r|x)$$

The Cold Start KBP problem can be seen as finding the maximum a posteriori (MAP) configuration:

$$(u^*, y^*) = \arg \max_{(u,y)} Pr(u, y|x)$$

The baseline system approximates the solution by solving in 2 separate stages: solve EDL by fixing $u_m^* = \arg \max_{1 \leq i \leq |E|} \Psi_m^{EDL}(u_m = i|x_m)$ for all x_m , then solve RE by fixing $y_{i,j}^{*r} = \arg \max_{\delta \in \{0,1\}} \Psi_{i,j,r}^{RE}(y_{i,j}^r = \delta|x, u^*)$ for all i, j, r . The potential $\Psi_{i,j,r}^{RE}(y_{i,j}^r = 1|x, u^*)$ is a relaxed form for $\Psi_{i,j,r}^{RE}(y_{i,j}^r = 1|x)$:

$$\begin{aligned} & \Psi_{i,j,r}^{RE}(y_{i,j}^r = 1|x, u^*) = \\ & \max_{\substack{m,n: \\ (u_m^*, u_n^*) = (i,j)}} (\Psi_m^{EDL}(u_m^* = i|x_m) \Psi_n^{EDL}(u_n^* = j|x_n)) \\ & \Psi_{m,n}^{MRE}(z_{m,n} = r|x_m, x_n) \end{aligned}$$

In the relaxed form, the optimization problem required for estimating the potential of $y_{i,j}^{*r}$ is limited to search only over pairs of mentions x_m, x_n which are now associated with the entities e_i, e_j , i.e. $u_m = i$ and $u_n = j$, instead of enumerating over all pairs of mentions. This can be done efficiently.

4 Incorporating World Knowledge

We incorporate world knowledge related to *entity popularity* and a set of *per-relation cardinalities* as additional factors in the objective. To learn these factors' form, we analyze real-world datasets and find that both factors follow the power-law distribution with long tails of low-frequency instances.

4.1 Entity Popularity

We define *entity popularity* (EP) as the number of mentions of an entity in a corpus. Entities vary in *popularity*— famous people (e.g. politicians, athletes), countries, and large organizations will be mentioned frequently in news, while other entities— a small city, the local valedictorian may only be mentioned a few times. Ideally, we would

model the EP distribution with counts from a large corpus annotated for names and cross-document entity coreference. As we are not aware of any such resource, we look at two approximations:

Name variants (NV): We collect name variants (e.g., *UN and United Nations*) for PER and ORG from Freebase and for GPE from GeoNames.

Name Mentions (NM): From a 50K document sample of Gigaword, for each entity, we search for exact matches to its name variants, and count these matches to estimate the number of entity mentions.

Figure 2 (left) plots the per-entity relationship between the count of NM and NV with rank. Both follow the power-law distribution (i.e. the plots are close to straight lines in a log-log scale). In other words, most entities have only a small number of variants and are mentioned only a few times. A handful of popular entities are mentioned frequently and have many variants⁸. The size of entities in the Kripke (Finin et al., 2014) system follows power-law distribution, further supporting our findings.

Formally, we define for i th entity the popularity variable $q_i(u) = \sum_m I(u_m = i)$ and the potential for EP factor $\Psi_i^{EP}(u)$ as follows:

$$\Psi_i^{EP}(u) \propto \exp(\theta^{EP}(q_i(u)))$$

in which $\theta^{EP}(q_i(u)) = \alpha \ln(q_i)$.

The parameter $\alpha > 0$ is initially fit from Freebase entities and then finetuned with TAC 2014 dataset to reflect real-world distributions. The EP term favors EDL solutions u with a popularity distribution that follows a power-law with a long tail of low-frequency entities.

4.2 Relation Cardinality

We define relation r 's *cardinality* regarding e_i as the number of entities or values associated with e_i through r . For example, if *John Smith* has 3 children, the cardinality of $r=per:children$ regarding $e_i=John\ Smith$ is 3. Formally, we notate the set of variables $\{y_{ij}^r\}$ as y_i^r , and the cardinality of a relation r of entity e_i as $d_i^r = \sum_j y_{ij}^r$.

Per-relation cardinalities (RC) often reflect real world constraints— people have at most one birthdate and typically no more than 5 siblings. To understand the cardinality constraints for the Cold Start relations, we use Freebase, a large, manually

⁸We confirm that a few entities have many variants e.g. for *Elizabeth II*, 364 variants including *Elizabeth II of the UK*, *Her Majesty the Queen*, *Queen of Australia*, etc.

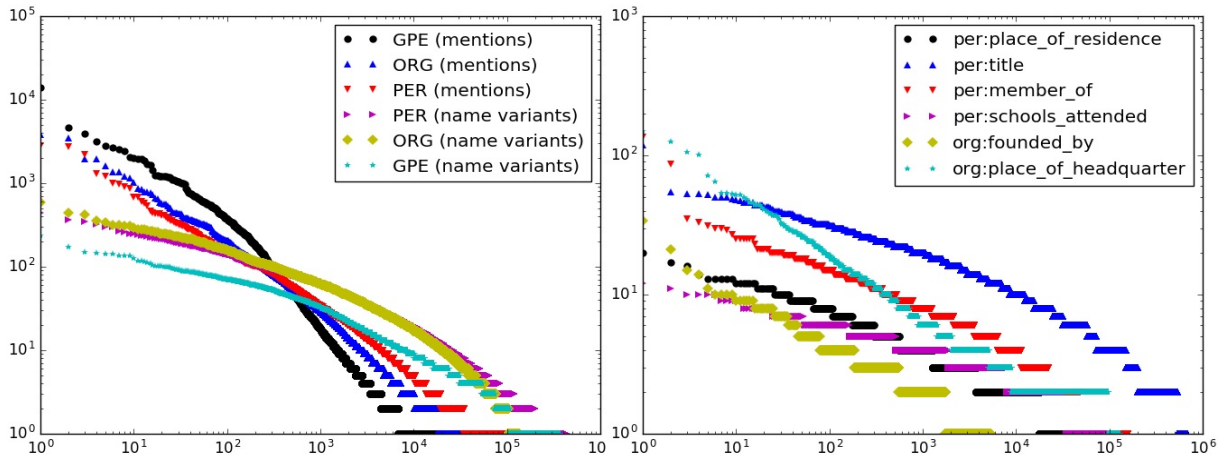


Figure 2: Real-world entity popularity (left) and per-relation cardinality (right) both follow the power-law distribution. x -axis shows ranks and y -axis shows counts (both are in log scale). The Left figure plots both numbers of name variants and numbers of mentions to ranks for PER, ORG and GPE.

curated KB. We align Freebase to the TAC schema following (Chen et al., 2010) and then generate a cardinality for each relation for all entities. The relationship between RC and RC-rank for the the 6 most frequent relations is plotted in Figure 2 (right). For these relations, RC closely resembles a power-law distribution. To favor both power-law and a soft size-limit on cardinality, we define the following potentials for RC factors for each relation-entity pair:

$$\Psi_{i,r}^{RC}(y^r) \propto \exp(\theta^{RC}(d_i^r))$$

in which $\theta^{RC}(d_i^r) = \beta \ln(d_i^r) - \gamma(\max(d_i^r - \mu^r, 0))^2$ with parameters $\beta, \gamma > 0$, and μ^r as the mean of the cardinalities of a relation r (estimated from Freebase). The first term in the potential has the power-law assumption, while the second term penalizes large cardinalities for going beyond the mean μ^r .

4.3 Incorporating Prior World Knowledge

Incorporating the EP and per-relation RC terms into the joint distribution, we obtain the joint objective:

$$Pr^*(u, y|x) \propto Pr(u, y|x) \cdot \prod_{1 \leq i \leq |E|} \Psi_i^{EP}(u) \cdot \prod_{\substack{1 \leq i \leq |E| \\ r \in R}} \Psi_{i,r}^{RC}(y_i^r)$$

with $Pr(u, y|x)$ as the baseline objective. A simplified plate diagram is shown in Figure 1.

Learning constraints for real-world corpora:

As we’re not aware of any large corpus annotated exhaustively with entities and relations, we fit the parameters of the constraints initially from

Freebase entities and relations, and then fine-tune them using empirical utility maximization (Jansche, 2005; Ye et al., 2012) for TAC Cold Start all-hop F1 with grid search in the parameter space, using previous years’ TAC assessment. Freebase is used in initialization because of its scale while finetuning with TAC assessment ensures the factors to more appropriately represent the underlying distribution of entity popularity and relation cardinality in a real-world corpus.

5 Jointly Inferring Entities and Relations

The problem of Cold Start KBP becomes finding a MAP assignment of u and y for $Pr^*(u, y|x)$. Finding the exact solution is hard, as many terms in the objective involve large groups of variables. We propose Algorithm 1 as an approximate heuristic. Line 1 generates an initial KB by approximating a solution for the baseline objective $Pr(u, y|x)$ (Section 3), but tends to overlink entities and over-aggregate relations. Lines 2-8 iteratively refine the KB by searching over the (u, y) -space using operation $o \in \{\text{SplitE}, \text{PruneR}\}$. At t -th iteration, it performs the operation o with the highest potential gain $\Delta \ln Pr^*(o(u^t, y^t)|x)$. The process is repeated until the gain is smaller than a very small value ϵ .

SplitE: splits an entity e_i into two entities. Since there are an exponential number of possible SplitE actions, we uses the following two heuristics: 1) cluster name mentions by their string forms, and find an “outlier” cluster of mentions, 2) rank e_i ’s mentions $\{x_m : u_m^* = i\}$ by their local EDL potential $\Psi_m^{EDL}(u_m^*|x_m)$ and find the lowest-ranked mention as an “outlier”. After find-

Input : $x, \alpha, \beta, \gamma, \mu^r$ for $r \in R$

Output: u, y

```

1  $(u^0, y^0) = \arg \max_{(u,y)} Pr(u, y|x)$ 
2  $t \leftarrow 0$ 
3 repeat
4    $o = \arg \max_o \Delta \ln Pr^*(o(u^t, y^t)|x)$ 
5   if  $o! = null$  then
6      $(u^{t+1}, y^{t+1}) \leftarrow Execute(o(u^t, y^t))$ 
7      $t \leftarrow t + 1$ 
8 until  $\Delta \ln Pr^*(o(u^t, y^t)|x) < \epsilon;$ 
Algorithm 1: The MAP inference algorithm

```

ing an ‘‘outlier’’ mention cluster of e_i , we divide it into two entities: e_g with the ‘‘core’’ mentions, and e_h with the ‘‘outlier’’ mention cluster. We repeat the process to find all outlier entities and separate them from the entity. Relation arguments will be reattached to the new entities accordingly. We only consider a short list of most popular entities and split each using the heuristics described above.

PruneR: removes a batch of relations ($Y = \{y_{i,j}^r\}$) by setting 0: $y_{i,j}^r \leftarrow 0$ for each $y_{i,j}^r \in Y$. The batch is generated with the following steps: first select a set of entity-relation pairs (e_i, r) with the highest cardinality $d_i^r = \sum_j y_{i,j}^r$, then repeatedly select the associated relation with the lowest potential $j^* = \arg \min_{j: y_{i,j}^r=1} \Psi^{RE}(y_{i,j}^r|x)$. Each $y_{i,j}^r$ will be added into the batch until its size reaches 50.

We define the gain for SplitE and PruneR as:

$$\begin{aligned} \Delta \ln Pr^*(SplitE(e_g, e_h \leftarrow e_i)|x) = & \\ & \sum_m (\ln \Psi_m^{EDL}(g) + \ln \Psi_m^{EDL}(h) - \ln \Psi_m^{EDL}(i)) \\ & + \sum_{r \in R} (\ln \Psi_g^{RE} + \ln \Psi_h^{RE} - \ln \Psi_i^{RE}) \\ & + \ln \Psi_r^{RC}(y^r) - \ln \Psi_r^{RC}(y^r) \\ & + \ln \Psi_g^{EP}(u') + \ln \Psi_h^{EP}(u') - \ln \Psi_i^{EP}(u) \end{aligned}$$

$$\begin{aligned} \Delta \ln Pr^*(PruneR(Y)|x) = & \\ & \sum_{r \in R} (\ln \Psi_r^{RC}(y^r) - \ln \Psi_r^{RC}(y^r)) \\ & + \sum_{y_{i,j}^r \in Y} (\ln \Psi_0^{RE} - \ln(\Psi_{i,j,r}^{RE}(y_{ij}^r = 1|x))) \end{aligned}$$

in which $\Psi_m^{EDL}(i)$ is short for $\Psi_m^{EDL}(u_m = i|x_m)$ with m ranges over IDs of mentions in e_i , and Ψ_i^{RE} is the sum of the RE factors which are related to entity e_i . y', u' are the assignment to y, u if a SplitE or a PruneR operation is executed. We also use the short form $\Psi_r^{RC}(y^r)$ as the sum of the RC factors which have changed because of a SplitE or a PruneR operation.

Since the gain is only computed for the short-listed entities and relations, and we only calcu-

late the subset of factors (EDL, RE, RC, and EP) related to the operation, $\Delta \ln Pr^*(SplitE|x)$ and $\Delta \ln Pr^*(PruneR|x)$ can be calculated efficiently.

6 Experiments

We evaluate our system with resources provided to TAC 2015 participants, including 1) a source corpus of 50,000 documents from newswire and discussion forums, 2) a query set consisting of 317 hop-0 entities (expanded to 1,148 hop-0 entry-point mentions and 8,191 hop-1 queries), 3) LDC⁹ assessment of participant responses from automatic submissions¹⁰ and a manually created submission¹¹, and 4) software that retrieves answers from a KB and measure performance with the assessment. Additionally, we use TAC 2013 and 2014 datasets for tuning parameters and training stacked classifiers. $\alpha = 10, \beta = 5, \gamma = 0.1$ are set empirically following Section 4.3. We run each experiment 20 times and average the scores.

6.1 Queries, Assessment, and Scoring

We briefly describe the evaluation process and scoring metrics. More details appear in (Mayfield, 2014). The Cold Start evaluation measures KB-quality by probing the KB with two types of queries. The queries are either at **hop-0** (e.g., *which organization(s) is(are) founded by Bill Gates?*) or **hop-1** (e.g., *in which city(-ies) the organization(s) founded by Bill Gates is(are) headquartered?*). More formally, the evaluation software tries to find an entity e_0 in the submitted KB that covers the entry-point mention of a hop-0 query q_0 , then finds all relational triples matching $(e_0, r_1, ?)$. X , the set of entities matching the open variable, is reviewed by annotators for: (a) assessment of correctness and (b) the identification of non-redundant subset X' . The software generates an hop-1 query $q_1 = x'$ for each $x' \in X'$, finds the entity e_1 that aligns with q_1 , and then finds triples matching $(e_1, r_2, ?)$. This results in response set Y , the set of entities matching the second open variable. Set Y is assessed by LDC in the same manner as Set X . The process is performed over all submitted KBs¹². The answers in X (hop-0)

⁹<https://www ldc.upenn.edu/>

¹⁰20 teams participants in the task with >50 KBs submitted.

¹¹created by time-limited LDC annotators

¹²LDC generates the time-limited manual run directly from text queries, but treats the responses identically for assessment

Systems	CS-SF								CS-LDC-MAX							
	Hop-0				Hop-1				Hop-0				Hop-1			
	P	R	F1	Ign	P	R	F1	Ign	P	R	F1	Ign	P	R	F1	Ign
TAC rank1	48	30	37	0	31	17	22	0	50	35	40	0	28	20	23	0
offset-based	50	31	38	64	31	18	23	23	52	35	42	19	30	21	24	6
string-match	49	31	38	13	32	19	24	11	52	36	42	6	29	21	25	3
assess	50	31	39	0	32	19	23	0	53	37	43	0	29	21	24	0

Table 2: CS-SF and CS-LDC-Max micro-averaged precision, recall and F1 of hop-0 and hop-1 queries for the TAC 2015 top-ranked KB submission (Min et al., 2015) and our KBC+E+R system (3 bottom rows) using various post-hoc scoring techniques (offset-based, string-match and assess). *Ign* is the number of unassessed answers.

are correct when sufficiently justified in the source corpus. The answers in Y (hop-1) are correct only if both the element of X that generated the response is correct and the response in Y is justified in the text.

NIST reports two metrics, **CS-SF** and **CS-LDC-MAX**, which differ in the treatment of multiple entry-point mentions for a single real-world entity. CS-SF treats each distinct mention as an independent query. CS-LDC-MAX takes only the entry-point mention which maximizes system performance for a given query-entity (i.e. either the responses for *Bill Gates* or *William Gates*). For both metrics, NIST calculates micro-averaged precision, recall, and F1 over all queries. As mentioned above, the official evaluation is a human post-hoc assessment of KB output. A system developed outside of the evaluation window, e.g., our proposed algorithm, will likely include responses for which truth is not known, which are ignored by the scoring software. Table 2 compares the TAC top-ranked system to our full configuration using three post-hoc scoring strategies: strict **offset-based** match, **string-match** match, and **assess** in which we apply the offset-based metric using additional internally performed assessments. For the ablation study in Table 3, we use the official scorer’s **string-match** mode. A small number of responses are ignored (**Ign**) even in string-match mode. We further account for these responses by re-estimating precision for hop-0 and hop-1 assuming that the precision of the ignored responses at hop-1 is the same as the hop-0 precision¹³. When this optimistic estimate differs from reported precision, we report it in parentheses.

¹³This overestimates the hop-1 precision which is lower than hop-0 precision because of error compounding.

6.2 Results and Discussion

Table 2 compares our full system (KBC+E+R) to the top performing system in TAC Cold Start 2015 using three different approaches to post-hoc scoring. Without manual effort, our joint modeling approach exceeds the performance of the top-ranked system, which uses a cascade of manually-specified rules (Min et al., 2015). Our system obtains 5.4% and 4.8% relative improvement in hop-0 and hop-1 CS-SF F1 over the top-ranked system. Improvement is observed in both hop-0 and hop-1 and with both CS-SF and CS-LDC-MAX showing that the improvement is robust. A sign test shows that the improvements are significant with $p < 0.01$.

Systems	Hop-0			Hop-1		
	P	R	F1	P	R	F1
KBC	45	33	37.9	14(16)	21	17
KBC+E	45	32	37.9	18(21)	21	19
KBC+R	49	32	38.2	27	19	23
KBC+E+R	49	31	38.4	32	19	24

Table 3: CS-SF scores (string-match) for different priors: KBC (baseline: no world knowledge), KBC+E (only entity-based factors), KBC+R (only relation-based factors), KBC+E+R (both sets of factors). Numbers in parentheses indicate the optimistic estimate when it differs from the number reported by the scoring software.

Table 3 ablates each type of world knowledge to show the impact of entity and relation-based factors independently when compared to a version of our system without world knowledge. As expected, the impact of world knowledge is seen in improvements in precision at minor costs to recall. Both types of world knowledge have higher impact on hop-1 than hop-0 as hop-1 measures the formation of the KB with multiple hops in relations. Adding the relation factors has a larger

impact than adding the entity factors because our splitting of entities is conservative (only affects $< 0.1\%$ entities) while relations' factors removes 7.3% relations. The two classes of factors appear to have largely independent impacts—combining them yields a large improvement. In total, adding prior world knowledge yields relative improvements of 9% in hop-0 precision, 131% on hop-1 precision, 42% on hop-1 F1, and 19.4% on all-hop F1 over the baseline. A sign test shows the improvements are significant with $p < 0.01$.

Reduction of errors: With relation factors added (KBC+R and KBC+E+R), 7.3% relations (out of 243K) are removed by **PruneR** with minimal recall loss. The median number of fillers for relations for the top 1% entities drops, e.g. *per:title* from 7 to 5, *per:employee_or_member_of* from 5 to 2, and *per:city_of_birth* from 3 to 1. Inspection shows that our approach addresses many obvious mistakes: *U.K.* is removed as a response to (*Securities and Exchange Commission(SEC)*, *org:country_of_headquarters*, ?) while *U.S.* remains. The error, caused by *UK's SEC* which means *UK's* analog to the *SEC* of *US*, is very hard to resolve without world knowledge. With cardinality constraints that favor only one *country_of_headquarters* for an ORG and *U.K.* has a lower confidence than *U.S.* as a filler, the model identifies *U.K.* as an incorrect answer.

With entity factors added (KBC+E+R and KBC+E), the model favors a larger amount of smaller but more precise entities. It generates 4% new entities (out of 212K) by splitting the largest $< 0.1\%$ entities with the **SplitE** heuristics described in section 5. For example, the entity *Australia* is splitted into 3 entities, *Australia* and two outliers *West Aussie* and *Australian Capital Territory*. It also singles out entities such as *South America*, *Idaho*, *Colorado* from the giant *U.S.* entity with $> 20,000$ mentions. When querying the KB facts related to *U.S.*, erroneous answers that would otherwise be reported through relations associated with *South America* or the *U.S.* states will be removed.

7 Related Work

Cold Start KBP The TAC Cold Start KBP workshop has attracted many text-based KBP systems (McNamee et al., 2012; McNamee et al., 2013; Mayfield et al., 2014; Min et al., 2015; Roth et al., 2015; Angeli et al., 2014; Nguyen et al., 2014; Monahan and Carpenter, 2012).

KELVIN (Mayfield et al., 2014) and BBN system (Min et al., 2015) both use hand-crafted rules to limit the number of fillers, e.g., remove less precise relations if a *person* has more than 8 (current and ex-) spouses. (Wolfe et al., 2015) and (He and Grishman, 2015) proposed interactive tools for KB construction with human guidance.

Knowledge Base Completion With the recent popularity of structured KBs such as Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and above-mentioned KBP techniques, there is a growing interest in completing a partially-complete KB with tensor decomposition (Chang et al., 2014), matrix factorization (Riedel et al., 2013), graph random walk (Lao et al., 2011; Lao et al., 2012; Gardner et al., 2014), neural networks (Socher et al., 2013; Neelakantan et al., 2015; Dong et al., 2014) and others (Guu et al., 2015; Gardner et al., 2013; Das et al., 2016). Knowledge Vault (Dong et al., 2014) pushes it further by combining many extraction components while estimating the confidence of their extractions and scales it to the Web. Model combination (Viswanathan et al., 2015) and confidence estimation (Wick et al., 2013; Li and Grishman, 2013) is related to our model for combining extraction components. The work described here differs from KB completion tasks in its requirement that the initial KB is empty and that all information in the KB be grounded in a text corpus.

Joint Modeling and Inference for IE To address the problem of compounding errors with multiple NLP components for IE, several papers (Finkel and Manning, 2009; Mccallum and Jensen, 2003; Finkel et al., 2006; Yao et al., 2010; Singh et al., 2009; Poon and Domingos, 2007; Wellner et al., 2004; Poon and Vanderwende, 2010; Riedel and McCallum, 2011; Chen et al., 2014; Kate and Mooney, 2010; Miwa and Sasaki, 2014) propose joint modeling and inference for IE. (Roth and Yih, 2007) use the ILP framework to enforce manually-specified constraints between entity and relation identification, while (Yu and Lam, 2010) models these two tasks in encyclopedia articles using a discriminative probabilistic model. (Li and Ji, 2014) jointly extracts entity mentions and relations with a structured perception with beam search. (Singh et al., 2013a) performs joint inference for entity, relation and coreference with an extension of the belief propagation algorithm. The work described here differs in its use of world knowledge. The joint modeling and

inference for IE is not comparable but complementary to our method, therefore can be incorporated into our system for further gain.

8 Conclusion and Future Work

We present a joint probabilistic framework for end-to-end Cold Start KBP with prior world knowledge. Experiments show it surpassing the best-performing system at the NIST TAC 2015 Cold Start evaluation. We plan to investigate additional world knowledge in the near future.

Acknowledgments

The authors would like to thank Scott Miller, Ralph Weischedel, and the anonymous reviewers for their very helpful comments on improving the paper.

References

- Gabor Angeli, Sonal Gupta, Melvin Johnson Premkumar, Christopher D Manning, Re Christopher, Julie Tibshirani, Jean Y Wu, Sen Wu, and Ce Zhang. 2014. Stanford’s Distantly Supervised Slot Filling Systems for KBP 2014. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD ’08*, page 1247, New York, New York, USA. ACM Press.
- Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed Tensor Decomposition of Knowledge Bases for Relation Extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579, Doha, Qatar, oct. Association for Computational Linguistics.
- Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Javier Artilles, Matthew Snover, Marissa Passantino, and Heng Ji. 2010. CUNY-BLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Liwei Chen, Yansong Feng, Jinghui Mo, Songfang Huang, and Dongyan Zhao. 2014. Joint Inference for Knowledge Base Population. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1912–1923, Doha, Qatar, oct. Association for Computational Linguistics.
- Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2016. Incorporating Selectional Preferences in Multi-hop Relation Extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 18–23, San Diego, CA, jun. Association for Computational Linguistics.
- Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’14*, pages 601–610, New York, New York, USA. ACM Press.
- Tim Finin, Paul McNamee, Dawn Lawrie, James Mayfield, and Craig Harman. 2014. Hot stuff at cold start: HLTCOE participation at TAC 2014. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Jenny Rose Finkel and Christopher D Manning. 2009. Joint Parsing and Named Entity Recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334, Boulder, Colorado, jun. Association for Computational Linguistics.
- Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Sydney, Australia, jul. Association for Computational Linguistics.
- Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving Learning and Inference in a Large Knowledge-Base using Latent Syntactic Cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838, Seattle, Washington, USA, oct. Association for Computational Linguistics.
- Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406, Doha, Qatar, oct. Association for Computational Linguistics.
- Ralph Grishman. 2013. Off to a Cold Start: New York University’s 2013 Knowledge Base Population Systems. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing Knowledge Graphs in Vector Space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages

- 318–327, Lisbon, Portugal, sep. Association for Computational Linguistics.
- Yifan He and Ralph Grishman. 2015. ICE: Rapid Information Extraction Customization for NLP Novices. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 31–35, Denver, Colorado, jun. Association for Computational Linguistics.
- Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA, jun. Association for Computational Linguistics.
- Martin Jansche. 2005. Maximum Expected F-Measure Training of Logistic Regression Models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 692–699, Vancouver, British Columbia, Canada, oct. Association for Computational Linguistics.
- Heng Ji, David Westbrook, and Ralph Grishman. 2005. Using Semantic Relations to Refine Coreference Decisions. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 17–24, Vancouver, British Columbia, Canada, oct. Association for Computational Linguistics.
- Rohit J Kate and Raymond Mooney. 2010. Joint Entity and Relation Extraction Using Card-Pyramid Parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden, jul. Association for Computational Linguistics.
- Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random Walk Inference and Learning in A Large Scale Knowledge Base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK., jul. Association for Computational Linguistics.
- Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading The Web with Learned Syntactic-Semantic Inference Rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026, Jeju Island, Korea, jul. Association for Computational Linguistics.
- Xiang Li and Ralph Grishman. 2013. Confidence Estimation for Knowledge Base Population. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 396–401, Hissar, Bulgaria, sep. INCOMA Ltd. Shoumen, BULGARIA.
- Qi Li and Heng Ji. 2014. Incremental Joint Extraction of Entity Mentions and Relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, jun. Association for Computational Linguistics.
- James Mayfield, Paul McNamee, Craig Harman, Tim Finin, and Dawn Lawrie. 2014. KELVIN: Extracting Knowledge from Large Text Collections. In *AAAI Fall Symposium on Natural Language Access to Big Data*, Arlington, Virginia.
- James Mayfield. 2014. Cold Start Knowledge Base Population at TAC 2014. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Andrew McCallum and David Jensen. 2003. A Note on the Unification of Information Extraction and Data Mining using Conditional-Probability, Relational Models. In *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, Acapulco, Mexico.
- Paul McNamee, Veselin Stoyanov, James Mayfield, Tim Finin, Tan Xu, Douglas W. Oard, and Dawn Lawrie. 2012. HLT/COE Participation at TAC 2012: Entity Linking and Cold Start Knowledge Base Construction. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Paul McNamee, Tim Finin, Dawn Lawrie, and James Mayfield. 2013. HLT/COE Participation at TAC 2013. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Bonan Min, Marjorie Freedman, and Constantine Lignos. 2015. BBN’s 2015 System for Cold Start Knowledge Base Population. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, aug. Association for Computational Linguistics.
- Makoto Miwa and Yutaka Sasaki. 2014. Modeling Joint Entity and Relation Extraction with Table Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, oct. Association for Computational Linguistics.
- Sean Monahan and Dean Carpenter. 2012. Lorify: A Knowledge Base from Scratch. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional Vector Space Models for Knowledge Base Completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International*

- Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 156–166, Beijing, China, jul. Association for Computational Linguistics.
- Thien Huu Nguyen, Yifan He, Maria Pershina, Xiang Li, and Ralph Grishman. 2014. New york university 2014 knowledge base population systems. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. *LDC2011T07*.
- Hoifung Poon and Pedro Domingos. 2007. Joint inference in information extraction. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, pages 913–918, Vancouver, British Columbia, Canada. AAAI Press.
- Hoifung Poon and Lucy Vanderwende. 2010. Joint Inference for Knowledge Extraction from Biomedical Literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821, Los Angeles, California, jun. Association for Computational Linguistics.
- Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A Multi-Pass Sieve for Coreference Resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501, Cambridge, MA, oct. Association for Computational Linguistics.
- Lance Ramshaw, Elizabeth Boschee, Marjorie Freedman, Jessica MacBride, Ralph Weischedel, and Alex Zamanian. 2011. *SERIF language processing effective trainable language understanding*. Springer-Link : B{ü}cher. Springer New York.
- Sebastian Riedel and Andrew McCallum. 2011. Fast and Robust Joint Models for Biomedical Event Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12, Edinburgh, Scotland, UK., jul. Association for Computational Linguistics.
- Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Barcelona, Spain. Springer, Berlin, Heidelberg.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, jun. Association for Computational Linguistics.
- Dan Roth and Scott Wen-tau Yih, 2007. *Global Inference for Entity and Relation Identification via a Linear Programming Formulation*, pages 553–580. MIT Press, nov.
- Benjamin Roth, Nicholas Monath, David Belanger, Emma Strubell, Patrick Verga, and Andrew McCallum. 2015. Building knowledge bases with universal schema: Cold start and slot-filling approaches. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Sameer Singh, Karl Schultz, and Andrew McCallum, 2009. *Bi-directional Joint Inference for Entity Resolution and Segmentation Using Imperatively-Defined Factor Graphs*, pages 414–429. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013a. Joint inference of entities, relations, and coreference. In *The 3rd Workshop on Automated Knowledge Base Construction*, pages 1–6, New York, New York, USA. ACM Press.
- Sameer Singh, Limin Yao, David Belanger, Ari Kobren, Sam Anzaroot, Michael Wick, Alexandre Passos, Harshal Pandya, Jinho Choi, Brian Martin, and Andrew McCallum. 2013b. Universal Schema for Slot Filling and Cold Start: UMass IESL at TACKBP 2013. In *Text Analysis Conference*, Gaithersburg, Maryland.
- Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.
- Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web - WWW '07*, pages 697–706, New York, New York, USA. ACM Press.
- Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea, jul. Association for Computational Linguistics.
- Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond Mooney. 2015. Stacked Ensembles of Information Extractors for Knowledge-Base Population. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 177–187, Beijing, China, jul. Association for Computational Linguistics.

- Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *the 20th conference on Uncertainty in artificial intelligence*, pages 593–601, Banff, Canada. AUAI Press.
- Michael Wick, Sameer Singh, Ari Kobren, and Andrew McCallum. 2013. Assessing confidence of knowledge base content with an experimental study in entity resolution. In *Proceedings of the 2013 workshop on Automated knowledge base construction - AKBC '13*, pages 13–18, New York, New York, USA. ACM Press.
- Travis Wolfe, Mark Dredze, James Mayfield, Paul McNamee, Craig Harman, Tim Finin, and Benjamin Van Durme. 2015. Interactive Knowledge Base Population. *CoRR*, abs/1506.0.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective Cross-Document Relation Extraction Without Labelled Data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, Cambridge, MA, oct. Association for Computational Linguistics.
- Nan Ye, Kian M Chai, Wee S Lee, and Hai L Chieu. 2012. Optimizing F-measure: A Tale of Two Approaches. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 289–296, New York, NY, USA. ACM.
- Xiaofeng Yu and Wai Lam. 2010. Jointly Identifying Entities and Extracting Relations in Encyclopedia Text via A Graphical Model Approach. In *Coling 2010: Posters*, pages 1399–1407, Beijing, China, aug. Coling 2010 Organizing Committee.

Generalizing to Unseen Entities and Entity Pairs with Row-less Universal Schema

Patrick Verga, Arvind Neelakantan and Andrew McCallum

College of Information and Computer Sciences

University of Massachusetts Amherst

{pat, arvind, mccallum}@cs.umass.edu

Abstract

Universal schema predicts the types of entities and relations in a knowledge base (KB) by jointly embedding the union of all available schema types—not only types from multiple structured databases (such as Freebase or Wikipedia infoboxes), but also types expressed as textual patterns from raw text. This prediction is typically modeled as a matrix completion problem, with one type per column, and either one or two entities per row (in the case of entity types or binary relation types, respectively). Factorizing this sparsely observed matrix yields a learned vector embedding for each row and each column. In this paper we explore the problem of making predictions for entities or entity-pairs unseen at training time (and hence without a pre-learned row embedding). We propose an approach having no per-row parameters at all; rather we produce a row vector on the fly using a learned aggregation function of the vectors of the observed columns for that row. We experiment with various aggregation functions, including neural network attention models. Our approach can be understood as a natural language database, in that questions about KB entities are answered by attending to textual or database evidence. In experiments predicting both relations and entity types, we demonstrate that despite having an order of magnitude fewer parameters than traditional universal schema, we can match the accuracy of the traditional model, and more importantly, we can now make predictions about unseen rows with nearly the same accuracy as rows available at training time.

1 Introduction

Automatic knowledge base construction (AKBC) is the task of building a structured knowledge base (KB) of facts using raw text evidence, and often an initial seed KB to be augmented (Carlson et al., 2010; Suchanek et al., 2007; Bollacker et al., 2008). KBs generally contain entity type facts such as *Sundar Pichai IsA Person* and relation facts such as *CEO.Of(Sundar Pichai,*

Google). Extracted facts about entities, and their types and relations are useful for many downstream tasks such as question answering (Bordes et al., 2014) and semantic parsing (Berant et al., 2013; Kwiatkowski et al., 2013).

An effective approach to AKBC is universal schema, which predicts the types of entities and relations in a knowledge base (KB) by jointly embedding the union of all available schema types—not only types from multiple structured databases (such as Freebase or Wikipedia infoboxes), but also types expressed as textual patterns from raw text. This prediction is typically modeled as a matrix completion problem. In the standard formulation for relation extraction (Riedel et al., 2013), entity pairs and relations occupy the rows and columns of the matrix respectively (Figure 1a). Analogously in entity type prediction (Yao et al., 2013), entities and types occupy the rows and columns of the matrix respectively (Figure 1b). The row and column entries are represented as learned vectors with compatibility determined by a scoring function.

In its original form, universal schema can reason only about row entries and column entries explicitly seen during training. Unseen rows and columns observed at test time do not have a learned embedding. This problem is referred to as the *cold-start* problem in recommendation systems (Schein et al., 2002).

Recently Toutanova et al. (2015) and Verga et al. (2016) proposed ‘column-less’ versions of universal schema models that generalize to unseen column entries. They learn compositional pattern encoders to parameterize the column matrix in place of individual column embeddings. However, these models still do not generalize to unseen row entries.

In this work, we present a ‘row-less’ extension of universal schema that generalizes to unseen entities and entity pairs. Rather than representing each row entry with an explicit dense vector, we encode each entity or entity pair as aggregate functions over their observed column entries. This is beneficial because when new entities are mentioned in text documents and subsequently added to the KB, we can directly reason on the observed text evidence to infer new binary relations and entity types for the new entities. This avoids the cumbersome effort of re-training the whole model from scratch to learn embeddings for the new entities.

To construct the row representation, we compare various aggregation functions in our experiments. We consider query independent and dependent aggregation functions. We find that query dependent attentional models that selectively focus on relevant evidence outperform the query independent alternatives. The query dependent attention mechanism also helps in providing a direct connection between the prediction and its provenance. Additionally, our models have a much smaller memory footprint since they do not store explicit row representations.

It is important to note that our approach is different from sentence level classifiers that predict KB relations and entity types using a single sentence as evidence. First, we pool information from multiple pieces of evidence coming from both text and annotated KB facts, rather than considering a single sentence at test time. Second, our methods are not limited to a fixed schema but instead predict a richer set of labels (KB types and textual), enabling easier downstream processing closer to natural language interaction with the KB. Finally, our model gains additional training signal from multi-task learning of textual and KB types. Since universal schema leverages large amounts of unlabeled text we desire the benefits of entity pair modeling, and row-less universal schema facilitates learning entity pair representations without the drawbacks of the traditional one-embedding-per-pair approach.

The majority of current embedding methods for KB entity type prediction operate with explicit entity representations (Yao et al., 2013; Neelakantan and Chang, 2015) and hence, cannot generalize to unseen entities. In relation extraction, entity-level models (Nickel et al., 2011; García-Durán et al., 2016; Yang et al., 2015; Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Socher et al., 2013) can handle unseen entity pairs at test time. These models learn representations for the entities instead of entity pairs. Hence, these methods still cannot generalize to predict relations between an entity pair if even one of the entities is unseen. Moreover, Toutanova et al. (2015) and Riedel et al. (2013) observe that the entity pair model outperforms entity models in cases where the entity pair was seen at training time.

Most similar to this work, Neelakantan et al. (2015) classify KB relations by finding the maximum scoring path between two entities. This model is also ‘row-less’ and does not directly model entities or entity pairs. There are several important differences in this work. Neelakantan et al. (2015) learn per-relation classifiers to predict only a small set of KB relations, while we instead predict all relations, including textual relations. We also explore aggregation functions that pool evidence from multiple paths while Neelakantan et al. (2015) only chose the maximum scoring path. Additionally, we demonstrate that our models can perform on par with those with explicit row representations while Neelakantan et al. (2015) did not perform

this comparison.

In this paper we investigate universal schema models without explicit row representations on two tasks: entity type prediction and relation extraction. We use entity type and relation facts from Freebase (Bollacker et al., 2008) augmented with textual relations and types from Clueweb text (Orr et al., 2013; Gabrilovich et al., 2013). We explore multiple aggregation functions and find that an attention-based aggregation function outperforms several simpler functions and matches a model using explicit row representations with an order of magnitude fewer parameters. More importantly, we then demonstrate that our ‘row-less’ models accurately predict relations on unseen entity pairs and types on unseen entities.

2 Background: Universal Schema

Universal schema (Riedel et al., 2013; Yao et al., 2013) relation extraction and entity type prediction is typically modeled as a matrix completion task. In relation extraction, entity pairs and relations occupy the rows and columns of the matrix (Figure 1-a), while in entity type prediction, entities and types occupy the rows and columns of the matrix (Figure 1-b). During training, we observe some positive entries in the matrix and at test time, we predict the missing cells in the matrix. This is achieved by decomposing the observed matrix into two low-rank matrices resulting in embeddings for each column entry and each row entry. Test time prediction is performed using the learned low-rank column and row representations.

Let T be the training set consisting of examples of the form (r, c) , where row $r \in U$ and column $c \in V$, denote an entity pair and relation type in the relation extraction task, or entity and entity type in the entity type prediction task. Let $v(r) \in \mathbb{R}^d$ and $v(c) \in \mathbb{R}^d$ be the vector representations or embeddings of row $r \in U$ and column $c \in V$ that are learned during training. Given a positive example, $(r, c) \in T$ in training, the probability of observing the fact is given by,

$$P(y_{r,c} = 1) = \sigma(v(r) \cdot v(c)) \quad (1)$$

where $y_{r,c}$ is a binary random variable that is equal to 1 when (r, c) is a fact and 0 otherwise, and σ is the sigmoid function. The embeddings are learned using Bayesian Personalized Ranking (BPR) (Rendle et al., 2009) in which the probability of the observed triples are ranked above unobserved triples.

3 Model

In this section, we describe the model, discuss the different aggregation functions and give details on the training objective.

3.1 ‘Row-less’ Universal Schema

While column-less universal schema addresses reasoning over arbitrary textual patterns, it is still limited to

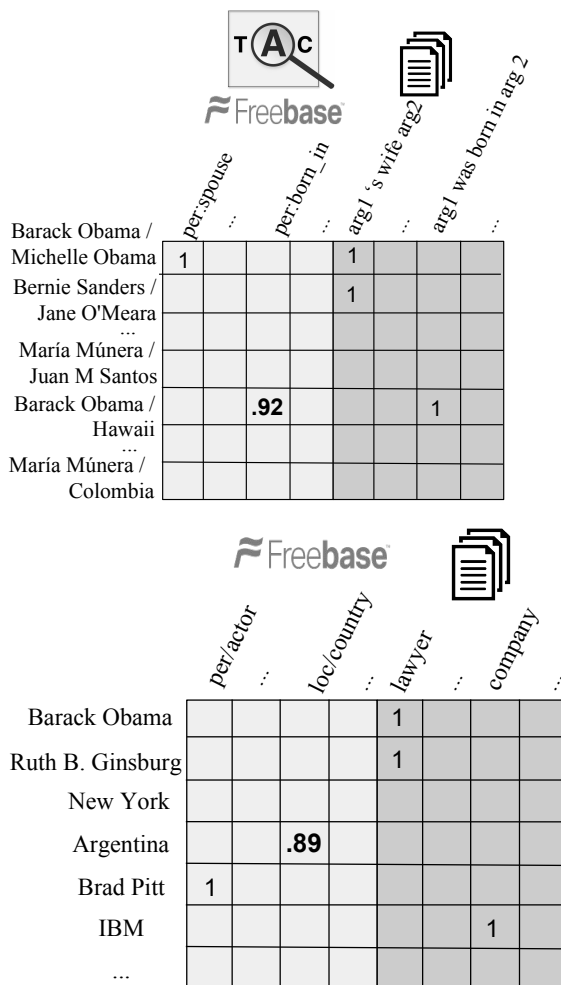


Figure 1: Universal schema matrix. a: Relation extraction. Relation types are represented as columns and entity pairs as rows of a matrix. Both KB relation types and textual patterns from raw text are jointly embedded in the same space. b: Entity type prediction. Entity types are represented as columns and entities as rows of a matrix.

reasoning over row entries seen at training time. Verga et al. (2016) use column-less universal schema for relation extraction. They address the problem of unseen row entries by using universal schema as a sentence classifier – directly comparing a textual relation to a KB relation to perform relation extraction. However, this approach is unsatisfactory for two reasons. The first is that this creates an inconsistency between training and testing. The model is trained to predict compatibility between rows and columns, but at test time it predicts compatibility between relations directly. Second, it considers only a single piece of evidence in making its prediction.

We address both of these concerns in our ‘row-less’ universal schema. Rather than explicitly encoding each row, we encode the row as a learned aggregation over their observed columns (Figure 2). A row contains an

entity for type prediction and an entity pair for relation extraction while a column contains a relation type for relation extraction and an entity type for type prediction. A learned row embedding can be seen as a summarization of all columns observed with that particular row. Instead of modeling this summarization as a single embedding, we reconstruct a row representation from an aggregate of its column embeddings, essentially learning a mixture model rather than a single centroid.

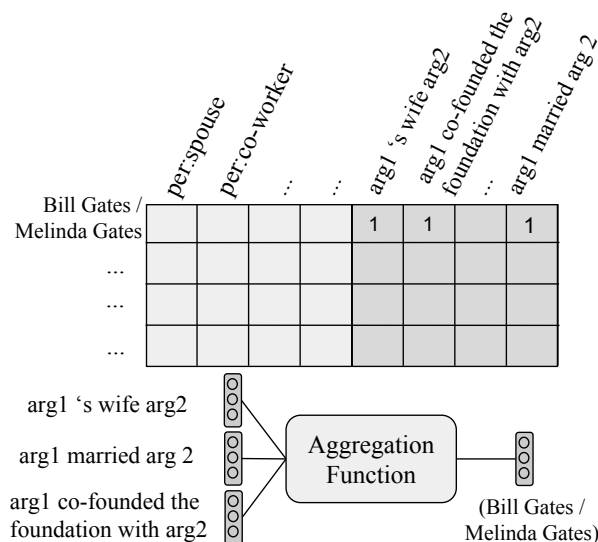


Figure 2: Row-less universal schema for relation extraction encodes an entity pair as an aggregation of its observed relation types.

3.2 Aggregation Functions

In this work we examine four aggregation functions to construct the representations for the row. Let $v(\cdot)$ denote a function that returns the vector representation for rows and columns. To model the probability between row r and column c , we consider the set $V(r)$ which contains the set of column entries that are observed with row r at training time, i.e.,

$$\forall \bar{c} \in V(r), (r, \bar{c}) \in T$$

The first two aggregation functions create a single representation for each row independent of the query. **Mean Pool** creates a single centroid for the row by averaging all of its column vectors,

$$v(r) = \sum_{\bar{c} \in V(r)} v(\bar{c})$$

While this formulation intuitively makes sense as an approximation for the explicit row representation, averaging large numbers of embeddings can lead to a noisy representation.

Max Pool also creates a single representation for the row by taking a dimension-wise max over the observed column vectors:

$$v(r)_i = \max_{\bar{c} \in V(r)} v(\bar{c})_i, \forall i \in 1, 2, \dots, d$$

where a_i denotes the i^{th} dimension of vector a . Both mean pool and max pool are query-independent and form the same representation for the row regardless of the query relation.

We also examine two query-specific aggregation functions. These models are more expressive than a single vector forced to act as a centroid to all possible columns observed with that particular row. For example, the entity pair Bill and Melinda Gates could hold the relation ‘per:spouse’ or ‘per:co-worker’. A query-specific aggregation mechanism can produce separate representations for this entity pair dependent on the query.

The **Max Relation** aggregation function represents the row as its most similar column to the query vector of interest. Given a query relation c ,

$$c_{max} = \operatorname{argmax}_{\bar{c} \in V(\bar{r})} v(\bar{c}) \cdot v(c) \\ v(r) = v(c_{max})$$

A similar strategy has been successfully applied in previous work (Weston et al., 2013; Neelakantan et al., 2014; Neelakantan et al., 2015) for different tasks. This model has the advantage of creating a query-specific entity pair representation, but is more susceptible to noisy training data as a single incorrect piece of evidence could be used to form a prediction.

Finally, we look at an **Attention** aggregation function over columns (Figure 3) which is similar to a single-layer memory network (Sukhbaatar et al., 2015). The *soft attention* mechanism has been used to selectively focus on relevant parts in many different models (Bahdanau et al., 2015; Graves et al., 2014; Neelakantan et al., 2016).

In this model the query is scored with an input representation of each column embedding followed by a softmax, giving a weighting over each relation type. This output is then used to get a weighted sum over a set of output representations for each column resulting in a query-specific vector representation of the row. Given a query relation c ,

$$\operatorname{score}_{\bar{c}} = v(c) \cdot v(\bar{c}), \forall \bar{c} \in V(\bar{r}) \\ p_{\bar{c}} = \frac{\exp(\operatorname{score}_{\bar{c}})}{\sum_{\bar{c} \in V(\bar{r})} \exp(\operatorname{score}_{\bar{c}})}, \forall \bar{c} \in V(\bar{r}) \\ v(r) = \sum_{\bar{c} \in V(\bar{r})} p_{\bar{c}} \times v(\bar{c})$$

The model pools relevant information over the entire set of observed columns and selects the most salient aspects to the query.

Model	Parameters
Entity Embeddings	3.7 e6
Attention	3.1 e5
Mean Pool/Max Pool/Max Relation	1.5 e5

Table 1: Number of parameters for the different models on the entity type dataset.

3.3 Training

The vector representation of the rows and the columns are the parameters of the model. Riedel et al. (2013)

use Bayesian Personalized Ranking (BPR) (Rendle et al., 2009) to train their universal schema models. BPR ranks the probability of observed triples above unobserved triples rather than explicitly modeling unobserved edges as negative. Each training example is an (entity pair, relation type) or (entity, entity type) pair observed in the training text corpora or KB.

Rather than BPR, Toutanova et al. (2015) use 200 negative samples to approximate the negative log likelihood¹. In our experiments, we use the sampled approximate negative log likelihood which outperformed BPR in early experiments.

Each example in the training procedure consists of a row-column pair observed in the training set. For a positive example $(r, c) \in T$, we construct the set $V(\bar{r})$ containing all the other column entries apart from c that are observed with row r .

To make training faster and more robust, we add ‘pattern dropout’ for entity pairs with many mentions. We set $V(\bar{r})$ to be m randomly sampled mentions for entity pairs with greater than m total mentions. In our experiments we set $m = 10$ and at test time we use all mentions. We then use $V(\bar{r})$ to obtain the aggregated row representation as discussed above.

We randomly sample 200 columns unobserved with row r to act as the negative samples. All models are implemented in Torch² and are trained using Adam (Kingma and Ba, 2015) with default momentum related hyperparameters.

4 Related Work

Relation extraction for KB completion has a long history. Mintz et al. (2009) train per relation linear classifiers using features derived from the sentences in which the entity pair is mentioned. Most of the embedding-based methods learn representations for entities (Nickel et al., 2011; Socher et al., 2013; Bordes et al., 2013) whereas Riedel et al. (2013) use entity pair representations.

‘Column-less’ versions of Universal Schema have been proposed (Toutanova et al., 2015; Verga et al., 2016). These models can generalize to column entries unseen at training by learning compositional pattern encoders to parameterize the column matrix in place of embeddings. Most of these models do not generalize to unseen entity pairs and none of them generalize to unseen entities. Recently, Neelakantan et al. (2015) introduced a multi-hop relation extraction model that is ‘row-less’ having no explicit parameters for entity pairs and entities.

Entity type prediction at the individual sentence level has been studied extensively (Pantel et al., 2012; Ling

¹Many past papers restrict negative samples to be of the same type as the positive example. We simply sample uniformly from the entire set of row entries

²data and code available at <https://github.com/patverga/torch-relation-extraction/tree/rowless-updates>

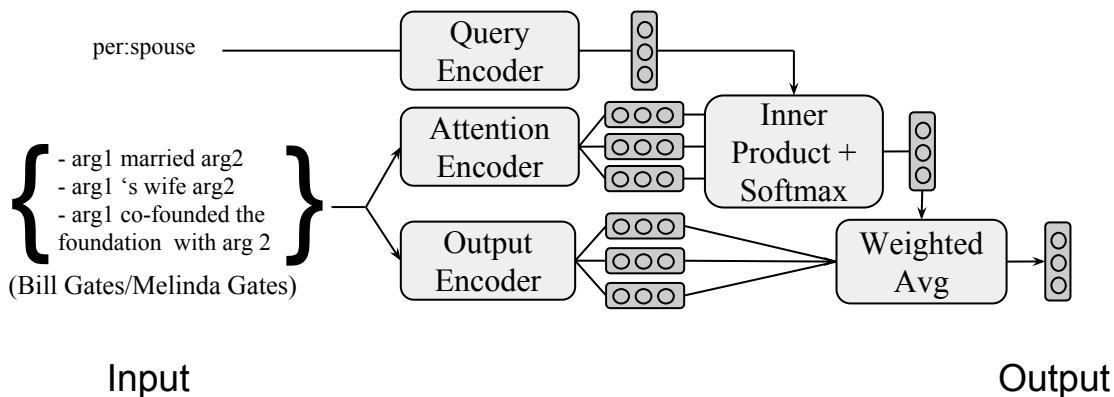


Figure 3: Example attention model in a row-less universal schema relation extractor. In the attention model, we compute the dot product between the representation of the query relation and the representation of an entity pair’s observed relation type followed by a softmax, giving a weighting over the observed relation types. This output is then used to get a weighted sum over the set of representations of the observed relation types. The result is a query-specific vector representation of the entity pair. The Max Relation model takes the most similar observed relation’s representation.

and Weld, 2012; Shimaoka et al., 2016). More recently, embedding-based methods for knowledge base entity type prediction have been proposed (Yao et al., 2013; Neelakantan and Chang, 2015). These methods have explicit entity representations, hence cannot generalize to unseen entities.

The task of generalizing to unseen row and column entries is referred to as the *cold-start* problem in recommendation systems. Methods proposed to tackle this problem commonly use user and item content and attributes (Schein et al., 2002; Park and Chu, 2009).

Multi-instance learning can be viewed as the relation classifier analogy of rowless universal schema. Riedel et al. (2010) used a relaxation of distant supervision training where all sentences for an entity pair (bag) are considered jointly and only the most relevant sentence is treated as the single training example for the bag’s label. Surdeanu et al. (2012) extended this idea with multi-instance multi-label learning (MIML) where each entity pair / bag can hold multiple relations / labels. Recently Lin et al. (2016) used a selective attention over sentences in MIML.

Concurrent to our work, Weissenborn (2016) proposes a row-less method for relation extraction considering both a uniform and weighted average aggregation function over columns. However, Weissenborn (2016) did not experiment with max and max-pool aggregation functions or evaluate on entity-type prediction. They also did not combine the rowless model with an LSTM column-less parameterization and did not compare to a model with explicit entity-pair representations.

5 Experimental Results

In this section, we compare our models that have aggregate row representations with models that have explicit row representations on entity type prediction and relation extraction tasks. Finally, we perform experiments on a column-less universal schema model. Table 1 shows that the row-less models require far fewer parameters since they do not explicitly store the row representations.

5.1 Entity Type Prediction

We first evaluate our models on an entity type prediction task. We collect all entities along with their types from a dump of Freebase³. We then filter all entities with less than five Freebase types leaving a set of 844780 (entity, type) pairs. Additionally, we collect 712072 textual (entity, type) pairs from Clueweb. The textual types are the 5000 most common appositives extracted from sentences mentioning entities. This results in 140513 unique entities, 1120 Freebase types, and 5000 free text types.

All embeddings are 25 dimensions, randomly initialized. We tune learning rates from $\{.01, .001\}$, ℓ_2 from $\{1e-8, 0\}$, batch size $\{512, 1024, 2048\}$ and negative samples from $\{2, 200\}$.

For evaluation, we split the Freebase (entity, type) pairs into 60% train, 20% validation, and 20% test. We randomly generate 100 negative (entity, type) pairs for each positive pair in our test set by selecting random entity and type combinations. We filter out false negatives that were observed true (entity, type) pairs in our complete data set. Each model produces a score for each positive and negative (entity, type) pair where the

³Downloaded March 1, 2015.

Model	MAP
Entity Embeddings	54.81
Mean Pool	39.47
Max Pool	32.59
Attention	55.66
Max Relation	55.37

(a)

Model	MAP
Entity Embeddings	3.14
Mean columns	34.77
Max column	43.20
Mean Pool	35.53
Max Pool	30.98
Attention	54.52
Max Relation	54.72

(b)

Table 2: Entity type prediction. Entity embeddings refers to the model with explicit row representations. Mean Columns and Max Column are equivalent to Mean Pool and Max Relation respectively (Section 3.2) but use the column embeddings learned during training of the Entity Embeddings model. b: Positive entities are unseen at train time.

type is the query. We then rank these predictions, calculate average precision for each of the types in our test set, and then use those scores to calculate mean average precision (MAP).

Table 2a shows the results of this experiment. We can see that the query dependent aggregation functions (Attention and Max Relation) performs better than the query independent functions (Mean Pool and Max Pool). The performance of models with query dependent aggregation functions which have far fewer parameters match the performance of the model with explicit entity representations.

We additionally evaluate our model’s ability to predict types for entities unseen during training. For this experiment, we randomly select 14000 entities and take all (entity, type) pairs containing those entities. We remove these pairs from our training set and use them as positive samples in our test set. We then select 100 negatives (entity, type) pairs per positive as above.

Table 2b shows the results of the experiment with unseen entities. There is very little performance drop for models trained with query dependent aggregation functions. The performance of the model with explicit entity representations is close to random.

5.1.1 Qualitative Results

A query specific aggregation function is able to pick out relevant columns to form a prediction. This is particularly important for rows that are not described easily by a single centroid such as an entity with several very different careers or an entity pair with multiple

highly varied relations. For example, in the first row in Table 3, for the query */baseball/baseball_player* the model needs to correctly focus on aspects like */sports/pro_athlete* and ignore evidence information like */tv/tv_actor*. A model that creates a single query-independent centroid will be forced to try and merge these disparate pieces of information together.

5.2 Relation Extraction

We evaluate our models on a relation extraction task using the FB15k-237 dataset from Toutanova et al. (2015). The data is composed of a small set of 237 Freebase relations and approximately 4 million textual patterns from Clueweb with entities linked to Freebase (Gabrilovich et al., 2013). In past studies, for each (subject, relation, object) test triple, negative examples are generated by replacing the object with all other entities, filtering out triples that are positive in the data set. The positive triple is then ranked among the negatives. In our experiments we limit the possible generated negatives to those entity pairs that have textual mentions in our training set. This way we can evaluate how well the model classifies textual mentions as Freebase relations. We also filter textual patterns with length greater than 35. Our filtered data set contains 2740237 relation types, 2014429 entity pairs, and 176476 tokens. We report the percentage of positive triples ranked in the top 10 amongst their negatives as well as the MRR scaled by 100.

Models are tuned to maximize mean reciprocal rank (MRR) on the validation set with early stopping. The entity pair model used a batch size 1024, $\ell_2 = 1e-8$, $\epsilon = 1e-4$, and learning rate 0.01. The aggregation models all used batch size 4096, $\ell_2 = 0$, $\epsilon = 1e-8$, and learning rate 0.01. Each use 200 negative samples except for max pool which performed better with two negative samples. The column vectors are initialized with the columns learned by the entity pair model. Randomly initializing the query encoders and tying the output and attention encoders performed better and all results use this method. All models are trained with embedding dimension 25.

Our results are shown in Table 4a. We can see that the models with query specific aggregation functions give the same results as models with explicit entity pair representations. The Max Relation model performs competitively with the Attention model which is not entirely surprising as it is a simplified version of the Attention model. Further, the Attention model reduces to the Max Relation model for entity pairs with only a single observed relation type. In our data, 64.8% of entity pairs have only a single observed relation type and 80.9% have 1 or 2 observed relation types.

We also explore the models’ abilities to predict on unseen entity pairs (Table 4b). We remove all training examples that contain a positive entity pair in either our validation or test set. We use the same validation and test set as in Table 4a. The entity pair model predicts

Query	Observed Columns
/baseball/baseball_player	/sports/pro_athlete , /sports/sports_award_winner, /tv/tv_actor, /people/measured_person, /award/award_winner, /people/person
/architecture/engineer	engineer , /book/author, /projects/project_focus, /people/person, sir
/baseball/baseball_player	baseman , /sports/pro_athlete, /people/measured_person, /people/person, dodgers, coach
/computer/computer_scientist	/education/academic , /music/group_member, /music/artist, /people/person
/business/board_member	/organization/organization_founder , /award/award_winner, /computer/computer_scientist, /people/person, president, scientist
/education/academic	/astronomy/astronomer , /book/author

Table 3: Each row corresponds to a true query entity type (left column) and the observed entity types (right column) for a particular entity. The maximum scoring observed entity type for each query entity type is indicated in bold. The other types are in no particular order. It can be seen that the maximum scoring entity types are interpretable.

Model	MRR	Hits@10
Entity-pair Embeddings	31.85	51.72
Mean Pool	25.89	45.94
Max Pool	29.61	49.93
Attention	31.92	51.67
Max Relation	31.71	51.94

(a)

Model	MRR	Hits@10
Entity-pair Embeddings	5.23	11.94
Mean Pool	18.10	35.76
Max Pool	20.80	40.25
Attention	29.75	49.69
Max Relation	28.46	48.15

(b)

Table 4: The percentage of positive triples ranked in the top 10 amongst their negatives as well as the mean reciprocal rank (MRR) scaled by 100 on a subset of the FB15K-237 dataset. All positive entity pairs in the evaluation set are unseen at train time. Entity-pair embeddings refers to the model with explicit row representations. b: Predicting entity pairs that are not seen at train time.

random relations as it is unable to make predictions on unseen entity pairs. The query-independent aggregation functions, mean pool and max pool, perform better than models with explicit entity pair representations. Again, query specific aggregation functions get the best results, with the Attention model performing slightly better than the Max Relation model.

The two experiments indicate that we can train relation extraction models without explicit entity pair representations that perform as well as models with explicit representations. We also find that models with query specific aggregation functions accurately predict relations for unseen entity pairs.

5.3 ‘Column-less’ universal schema

The original universal schema approach has two main drawbacks: similar textual patterns do not share statistics, and the model is unable to make predictions about

entities and textual patterns not explicitly seen at train time.

Recently, ‘column-less’ versions of universal schema to address some of these issues (Toutanova et al., 2015; Verga et al., 2016). These models learn compositional pattern encoders to parameterize the column matrix in place of direct embeddings. Compositional universal schema facilitates more compact sharing of statistics by composing similar patterns from the same sequence of word embeddings – the text patterns ‘lives in the city’ and ‘lives in the city of’ no longer exist as distinct atomic units. More importantly, compositional universal schema can thus generalize to all possible textual patterns, facilitating reasoning over arbitrary text at test time.

The column-less universal schema model generalizes to all possible input textual relations and the row-less model generalizes to all entities and entity pairs, whether seen at train time or not. We can combine these two approaches together to make an universal schema model that generalizes to unseen rows and columns.

The parse path between the two entities in the sentence is encoded with an LSTM model. We use a single layer model with 100 dimensional token embeddings initialized randomly. To prevent exploding gradients, we clip them to norm 10 while all the other hyperparameters are tuned the same way as before. We follow the same evaluation protocol from 5.2.

The results of this experiment with observed rows are shown in Table 5a. While both the MRR and Hits@10 metrics increase for models with explicit row representations, the row-less models show an improvement only on the Hits@10 metric. The MRR of the query dependent row-less models is still competitive with the model with explicit row representation even though they have far fewer parameters to fit the data.

6 Conclusion

In this paper we explore a row-less extension of universal schema that forgoes explicit row representations for an aggregation function over its observed columns. This extension allows prediction between all rows in new textual mentions – whether seen at train time or not – and also provides a natural connection to the provenance supporting the prediction. Our models also have

Model	MRR	Hits@10
Entity-pair Embeddings	31.85	51.72
Entity-pair Embeddings-LSTM	33.37	54.39
Attention	31.92	51.67
Attention-LSTM	30.00	53.35
Max Relation	31.71	51.94
Max Relation-LSTM	30.77	54.80

(a)

Model	MRR	Hits@10
Entity-pair Embeddings	5.23	11.94
Attention	29.75	49.69
Attention-LSTM	27.95	51.05
Max Relation	28.46	48.15
Max Relation-LSTM	29.61	54.19

(b)

Table 5: The percentage of positive triples ranked in the top 10 amongst their negatives as well as the mean reciprocal rank (MRR) scaled by 100 on a subset of the FB15K-237 dataset. Negative examples are restricted to entity pairs that occurred in the KB or text portion of the training set. Models with the suffix “-LSTM” are column-less. Entity-pair embeddings refers to the model with explicit row representations. b: Predicting entity pairs that are not seen at train time.

a smaller memory footprint.

In this work we show that an aggregation function based on query-specific attention over relation types outperforms query independent aggregations. We show that aggregation models are able to predict on par with models with explicit row representations on seen row entries with far fewer parameters. More importantly, aggregation models predict on unseen row entries without much loss in accuracy. Finally, we show that in relation extraction, we can combine row-less and column-less models to train models that generalize to both unseen rows and columns.

Acknowledgments

We thank Emma Strubell, David Belanger, and Luke Vilnis for helpful discussions and edits. This work was supported in part by the Center for Intelligent Information Retrieval and the Center for Data Science, and in part by DARPA under agreement number FA8750-13-2-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon, in part by Defense Advanced Research Agency (DARPA) contract number HR0011-15-2-0036, and in part by the National Science Foundation (NSF) grant number IIS-1514053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. Arvind Neelakantan is supported by a Google PhD fellowship in machine learning.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference for Learning Representations (ICLR)*, San Diego, California, USA.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.
- Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.
- Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and A. 2010. Toward an architecture for never-ending language learning. In *AAAI*, Atlanta, Georgia, USA.
- Evgeniy Gabilovich, Michael Ringgaard, and Amanag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). <http://lemurproject.org/clueweb09/FACC1/>.
- Alberto García-Durán, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. 2016. Combining two and three-way embedding models for link prediction in knowledge bases. *Journal of Artificial Intelligence Research*, 55(1):715–742, January.
- Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *arXiv preprint arxiv:1410.5401*.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR)*, San Diego, California, USA.
- Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October. Association for Computational Linguistics.

- Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2181–2187, Austin, Texas, US.
- Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany, August. Association for Computational Linguistics.
- Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto, Ontario, CA.
- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August. Association for Computational Linguistics.
- Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525, Denver, Colorado, May–June. Association for Computational Linguistics.
- Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, October. Association for Computational Linguistics.
- Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166, Beijing, China, July. Association for Computational Linguistics.
- Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural Programmer: Inducing latent programs with gradient descent. In *4th International Conference for Learning Representations (ICLR)*, San Juan, Puerto Rico.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, Bellevue, Washington, USA.
- Dave Orr, Amarnag Subramanya, Evgeniy Gabrilovich, and Michael Ringgaard. 2013. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. <http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html>.
- Patrick Pantel, Thomas Lin, and Michael Gamon. 2012. Mining entity types from query logs via user intent modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 563–571, Jeju Island, Korea, July. Association for Computational Linguistics.
- Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28, New York, NY, USA. ACM.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, Montreal, QC, Canada.
- Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.
- Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM.
- Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74, San Diego, CA, June. Association for Computational Linguistics.
- Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, Lake Tahoe, Nevada, USA.

- Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, Banff, Alberta, Canada. ACM.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, Montreal, QC, Canada.
- Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea, July. Association for Computational Linguistics.
- Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoi-fung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, September. Association for Computational Linguistics.
- Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California, June. Association for Computational Linguistics.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119, Quebec City, QC, Canada.
- Dirk Weissenborn. 2016. Embedding entity pairs through observed relations for knowledge base completion. Unpublished manuscript, OpenReview.
- Jason Weston, Ron J Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 65–68, Hong Kong, China, October. ACM.
- Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference for Learning Representations (ICLR)*, San Diego, California, USA.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 79–84, San Francisco, CA, USA, October.

Learning to Generate Product Reviews from Attributes

Li Dong[†], Shaohan Huang[‡], Furu Wei[‡], Mirella Lapata[†], Ming Zhou[‡] and Ke Xu[†]

[†]University of Edinburgh, Edinburgh, United Kingdom

[‡]Microsoft Research, Beijing, China

[†]Beihang University, Beijing, China

li.dong@ed.ac.uk, {fuwei, mingzhou}@microsoft.com,
buaahsh@gmail.com, mlap@inf.ed.ac.uk, kexu@nlsde.buaa.edu.cn

Abstract

Automatically generating product reviews is a meaningful, yet not well-studied task in sentiment analysis. Traditional natural language generation methods rely extensively on hand-crafted rules and predefined templates. This paper presents an attention-enhanced attribute-to-sequence model to generate product reviews for given attribute information, such as user, product, and rating. The attribute encoder learns to represent input attributes as vectors. Then, the sequence decoder generates reviews by conditioning its output on these vectors. We also introduce an attention mechanism to jointly generate reviews and align words with input attributes. The proposed model is trained end-to-end to maximize the likelihood of target product reviews given the attributes. We build a publicly available dataset for the review generation task by leveraging the Amazon book reviews and their metadata. Experiments on the dataset show that our approach outperforms baseline methods and the attention mechanism significantly improves the performance of our model.

1 Introduction

Nowadays, there are many popular online review sites (such as Amazon, and Yelp) that allow users to read and post reviews about books, electronics, restaurants, etc. The reviews are used to express opinions for different aspects of products, and have a wide variety of writing styles and different polarity strengths. As a result, much previous work has focused on how opinions are expressed in review data. For example, previous studies on

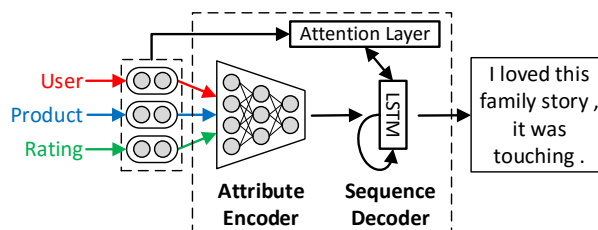


Figure 1: Our model learns to encode attributes into vectors, and then uses recurrent neural networks based on long short-term memory (LSTM) units to generate reviews by conditioning on the encoding vectors. An attention layer is used to learn soft alignments between attributes and generated words.

sentiment analysis identify and extract subjective content in review data (Liu, 2015). However, few studies have explored building data-driven models that can generate product reviews for the given products and ratings, which is helpful to understand how a specific user comments for products. As shown in Figure 1, the input to our model is a set of attributes (such as user, product, and rating information), and our goal is to generate user- and product-specific reviews that agree with the input rating. These automatically generated reviews are useful for companies. For example, we could promote a product to users who have not bought it, by generating novel and personalized recommendations. We could also build a review writing assistant for E-commerce websites. After the website generates some candidate reviews according to the user's rating score, users could select one and refine it, which makes the procedure more user-friendly. Moreover, we can generate novel and personalized recommendations for every user, which makes the recommendation system more interpretable.

This attribute-conditioned review generation

problem is very challenging due to the variety of candidate reviews that satisfy the input attributes. In other words, apart from the given attributes, there are other unknown or latent factors that influence the generated reviews, which renders the generation process non-deterministic. Moreover, although some attributes (such as rating) explicitly determine the usage of sentiment words, others (e.g., user information) implicitly influence word usage. So the model needs to handle both explicit and implicit clues. Additionally, the interactions between attributes are important to obtain the hidden factors used for generation. For example, different users tend to describe different aspects of a product and use different sentiment words to express a rating score.

In this paper, we propose a neural network based attribute-to-sequence model. As shown in Figure 1, our model contains three parts: attribute encoder, sequence decoder, and an attention mechanism. Specifically, we first use multilayer perceptrons to encode input attributes into vector representations that are used as latent factors for generating reviews. Next, the encoding vectors are fed into a coarse-to-fine sequence decoder. The decoder is built by stacking multiple layers of recurrent neural networks, which can generate words one by one conditioning on the encoding vectors. Besides, we introduce an attention layer into the proposed attribute-to-sequence model. The attention mechanism learns soft alignments between generated words and attributes, and adaptively computes encoder-side context vectors used to predict the next tokens. In order to evaluate our method, we build a dataset based on Amazon reviews and performed experiments on it. The experimental results show that the proposed model achieves superior performance against baseline methods. Moreover, we demonstrate that the attention mechanism significantly improves the performance of our model.

The contributions of this work are three-fold:

- We introduce the task of attribute-conditioned review generation, which is valuable for sentiment analysis, but not well studied previously.
- We propose an attention-enhanced attribute-to-sequence model in order to generate reviews conditioned on input attributes.
- We create a dataset based on Amazon book

reviews and present empirical studies to show the proposed model outperforms several baseline methods.

2 Related Work

Sentiment analysis and opinion mining aim to identify and extract subjective content in text (Liu, 2015). Most previous work focuses on using rule-based methods or machine learning techniques for sentiment classification, which classifies reviews into different sentiment categories. Recently, deep learning has achieved promising results on sentiment analysis (Socher et al., 2011; Dong et al., 2014; Kim, 2014). Lipton et al. (2015) use character-level concatenated input recurrent neural networks as a generative model to predict rating and category for reviews. In contrast, our model is mainly evaluated on the review generation task rather than classification. Moreover, we use an attention mechanism in our encoder-decoder model, which has been proved very helpful in various tasks (Bahdanau et al., 2015; Xu et al., 2015), to generate user- and product-specific reviews. Maqsd (2015) compare latent Dirichlet allocation, Markov chains, and hidden Markov models for text generation on review data. However, we focus on generating product reviews conditioned on input attributes. Park et al. (2015) propose to retrieve relevant opinion sentences using product specifications as queries, while we work on generation instead of retrieval.

Our task definition is also related to concept-to-text generation (Konstas and Lapata, 2012; Konstas and Lapata, 2013), such as generating weather forecast or sportscasting from database records. A typical system contains three main stages: content planning, sentence planning, and surface realization. Mei et al. (2016) treat database records and output texts as sequences, and use recurrent neural networks to encode and decode them. In contrast, our input is a set of discrete attributes instead of database records or sequences. In addition, the contents of database records are strong constraints on results in concept-to-text generation. However, in our setting, user and product information implicitly indicates the style of generated reviews, which makes the results extremely diverse.

Another line of related work is the encoder-decoder model with neural networks. Specifically, an encoder is employed to encode input information into vectors, and then a decoder learns to

predict results by conditioning outputs on the encoding vectors. This general framework is flexible because different neural networks can be used for encoders and decoders depending on the nature of inputs and outputs, which has been used to address various tasks. For example, recurrent neural networks are used to model sequences, such as machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014), syntactic parsing (Vinyals et al., 2015b), and semantic parsing (Dong and Lapata, 2016). Additionally, convolutional neural networks are employed for image data, such as image caption generation (Vinyals et al., 2015a), and video description generation (Donahue et al., 2015; Venugopalan et al., 2015). Our model employs multilayer perceptron to encode attribute information, and uses recurrent neural networks to decode product reviews. In order to better handle alignments between inputs and outputs, the attention mechanism is introduced for the encoder-decoder model. The attention model boosts performance for various tasks (Bahdanau et al., 2015; Luong et al., 2015; Xu et al., 2015). In our work, we use the attention mechanism to learn soft alignments between input attributes and output sequences, which has not, to our knowledge, been studied in previous work. Dosovitskiy et al. (2015) propose to use generative convolutional neural networks to generate images of chairs given chair type, viewpoint and color. Similarly, Yan et al. (2016) use variational auto-encoders to generate face images conditioned on visual attributes. However, our goal is to generate texts instead of images. Moreover, we learn a neural attention model to attend over input attributes during generation.

3 Modelling Approach

To begin with, we state the product review generation problem as follows. Given input attributes $a = (a_1, \dots, a_{|a|})$, our goal is to generate a product review $r = (y_1, \dots, y_{|r|})$ maximizing the conditional probability $p(r|a)$. Notice that number of attributes $|a|$ is fixed, while the review r is considered a word sequence of variable length. We use the user ID, product ID, and rating as attributes, so $|a|$ is set to 3 in our task. The training data are attributes paired with corresponding reviews. The model learns to compute the likelihood of generated reviews given input attributes. This condi-

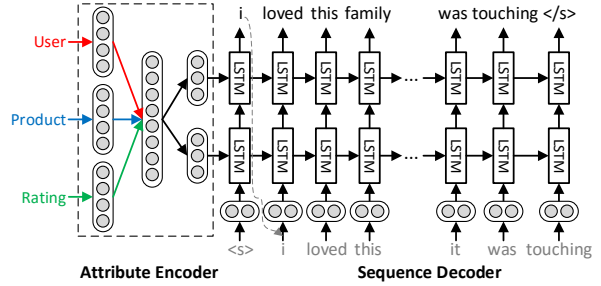


Figure 2: Attribute-to-sequence model without attention mechanism.

tional probability $p(r|a)$ is decomposed to:

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

where $y_{<t} = (y_1, \dots, y_{t-1})$.

Our method consists of three parts, i.e., an attribute encoder, a sequence decoder, and an attention layer. The attribute encoder employs multilayer perceptrons to encode attributes a to vectors. To be specific, we represent the attributes as vectors. Next, the concatenation of these vectors is fed into a hidden layer to obtain the encoding vectors. After we obtain the encoding vectors, the sequence decoder stacks L -layer recurrent neural networks (RNNs) to generate reviews conditioning on these vectors. During decoding, RNNs recurrently compute n -dimensional hidden vectors which are used to predict output words for different time steps. In order to better utilize encoder-side information, an attention layer is introduced to learn soft alignments between attributes and output words. For every decoding time step, we use the current hidden vector to compute attention scores over attribute vectors. Then, a weighted sum of attribute vectors is used as the context vector to predict output words.

We first describe the attribute-to-sequence model without using neural attention in Section 3.1 and Section 3.2. Next, we introduce the attention mechanism in Section 3.3.

3.1 Attribute Encoder

We use multilayer perceptrons with one hidden layer to encode attribute information into a vector as shown in Figure 2. At first, input attributes $a = (a_1, \dots, a_{|a|})$ are represented by low-dimensional vectors. The attribute a_i 's vector $\mathbf{g}(a_i)$ is computed via:

$$\mathbf{g}(a_i) = W_i^a \mathbf{e}(a_i) \quad (2)$$

where $W_i^a \in \mathbb{R}^{m \times |a_i|}$ is a parameter matrix, m is the dimension of embedding, and $\mathbf{e}(a_i) \in \{0, 1\}^{|a_i|}$ is a one-hot vector representing the presence or absence of a_i . Then, these attribute vectors are concatenated and fed into a hidden layer which outputs the encoding vector. The output of the hidden layer is computed as:

$$\mathbf{a} = \tanh(H[\mathbf{g}(a_1), \dots, \mathbf{g}(a_{|a|})] + \mathbf{b}_a) \quad (3)$$

where $[\mathbf{g}(a_1), \dots, \mathbf{g}(a_{|a|})]$ are concatenated attribute vectors, \tanh is a nonlinearity function, $H \in \mathbb{R}^{Ln \times |a|m}$ is a weight matrix, and $\mathbf{b}_a \in \mathbb{R}^{Ln}$ is the bias. Next, the vector \mathbf{a} is used to initialize the n -dimensional hidden vectors of the L -layer recurrent neural networks in the decoder.

3.2 Sequence Decoder

As shown in Figure 2, the decoder is built upon multilayer recurrent neural networks (RNNs) with long short-term memory (LSTM) units. RNNs use vectors to represent information for the current time step and recurrently compute the next hidden states. In our work, we stack multiple layers of RNNs in our architecture. Additionally, a long short-term memory (Hochreiter and Schmidhuber, 1997) unit is employed to better handle long sequences. The LSTM introduces several gates and explicit memory cells to memorize or forget information, which enables networks learn more complicated patterns. Let $\mathbf{h}_t^l \in \mathbb{R}^n$ denote an n -dimensional hidden vector in layer l and time step t . \mathbf{h}_t^l is computed via:

$$\mathbf{h}_t^l = f(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1}) \quad (4)$$

where $\mathbf{h}_t^0 = W^r \mathbf{e}(y_{t-1})$ is the word embedding of the previous predicted word, $W^r \in \mathbb{R}^{n \times |V_r|}$ is a parameter matrix, $|V_r|$ is the vocabulary size, and $\mathbf{e}(y_{t-1})$ is a one-hot vector used to extract word vector for y_{t-1} . We follow the architecture of LSTM unit described in Zaremba et al. (2015). To be specific, the unit is given by:

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^l \begin{pmatrix} \mathbf{h}_t^{l-1} \\ \mathbf{h}_{t-1}^l \end{pmatrix} \quad (5)$$

$$\mathbf{p}_t^l = \mathbf{f} \odot \mathbf{p}_{t-1}^l + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h}_t^l = \mathbf{o} \odot \tanh(\mathbf{p}_t^l)$$

where \tanh , sigm , and \odot are element-wise operators, and $W^l \in \mathbb{R}^{4n \times 2n}$ is a weight matrix for the l -th layer.

Once the input attributes are encoded to the vector $\mathbf{a} \in \mathbb{R}^{Ln}$ by Equation (3), the encoding vector is split into L vectors to initialize the hidden vectors of the first time step in decoder. Then, RNNs compute hidden vectors recurrently and predict output words using the hidden vectors of the top-most layer \mathbf{h}_t^L . For the vanilla model without using an attention mechanism, the predicted distribution of the t -th output word is:

$$p(y_t | y_{<t}, a) = \text{softmax}_{y_t}(W^p \mathbf{h}_t^L) \quad (6)$$

where $W^p \in \mathbb{R}^{|V_r| \times n}$ is a parameter matrix.

3.3 Attention Mechanism

The attention mechanism is introduced to better utilize encoder-side information. As indicated in Equation (6), the vanilla model does not directly use attribute vectors to generate sequences. Intuitively, the model can concentrate on different parts of encoding information to predict the next word. Previous work has proved this idea significantly improves performance especially for long sequences (Bahdanau et al., 2015; Vinyals et al., 2015b; Luong et al., 2015).

Figure 3 demonstrates how to compute the encoder-side context vector and use it to predict output words. For the t -th time step of the decoder, we compute the attention score of attribute a_i via:

$$s_i^t = \exp(\tanh(W^s[\mathbf{h}_t^L, \mathbf{g}(a_i)])) / Z \quad (7)$$

where the brackets $[\cdot, \cdot]$ denote concatenation, Z is a normalization term that ensures $\sum_{i=1}^{|a|} s_i^t = 1$, and $W^s \in \mathbb{R}^{1 \times (n+m)}$ is a parameter matrix. Next, the attention context vector \mathbf{c}^t is obtained by:

$$\mathbf{c}^t = \sum_{i=1}^{|a|} s_i^t \mathbf{g}(a_i) \quad (8)$$

which is a weighted sum of attribute vectors. We further employ the vector \mathbf{c}^t to predict the t -th output token as:

$$\mathbf{h}_t^{\text{att}} = \tanh(W_1 \mathbf{c}^t + W_2 \mathbf{h}_t^L) \quad (9)$$

$$p(y_t | y_{<t}, a) = \text{softmax}_{y_t}(W^p \mathbf{h}_t^{\text{att}}) \quad (10)$$

where $W^p \in \mathbb{R}^{|V_r| \times n}$, $W_1 \in \mathbb{R}^{n \times m}$ and $W_2 \in \mathbb{R}^{n \times n}$ are three parameter matrices.

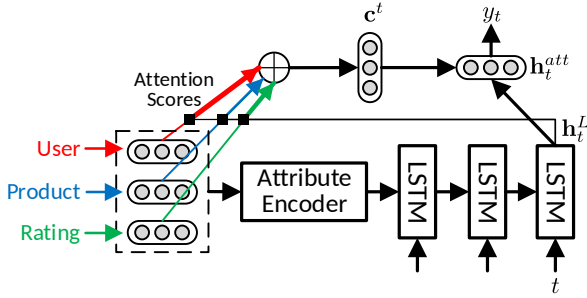


Figure 3: Attention scores are computed by attribute vectors and the current hidden vector of the decoder. Then, the encoder-side context vector is obtained in the form of a weighted sum, which is further used to predict the word distribution.

3.4 Model Training

We aim at maximizing the likelihood of generated reviews given input attributes for the training data. So we define the optimization problem as:

$$\text{maximize } \sum_{(a,r) \in \mathcal{D}} \log p(r|a) \quad (11)$$

where \mathcal{D} is the dataset of all attribute-review training pairs, and $p(r|a)$ is defined as shown in Equation (1). In order to avoid overfitting, we insert dropout layers between different LSTM layers as suggested in Zaremba et al. (2015). The mini-batched RMSProp (Tieleman and Hinton, 2012) algorithm is used to optimize the objective function.

3.5 Inference

At test time, we first use the encoder to encode input attributes into vectors, and use them to initialize the LSTM units of the decoder. Then, the decoder predicts a review \hat{r} that maximizes the conditional probability defined in Equation (1):

$$\hat{r} = \arg \max_{r'} p(r'|a) \quad (12)$$

where r' is a candidate review. Because we decompose this probability as shown in Equation (1), we can use beam search or greedy search to generate words, which avoids iterating over all candidate reviews. In order to determine the termination of the generation process, we add a special token $\langle /s \rangle$ to the end of every output review. The generation terminates once this token is emitted.

4 Experiments

We first introduce a new dataset for this task and compare our method with several baseline ap-

proaches. Then we conduct some ablation experiments and present model analysis to help us understand what the model learns.

4.1 Dataset Description

Our dataset is built upon Amazon product data (McAuley et al., 2015) that includes reviews and metadata spanning from May 1996 to July 2014 with duplicates removed. The products of the book domain are used in our experiments. Every review is paired with three attributes, i.e., user ID, product ID and rating. We filter books and users which do not occur at least 6 and 15 times, respectively. The reviews whose lengths are greater than 60 words are filtered. Because we observe that long reviews mainly describe the plots of books, while our goal is to generate reviews expressing opinions. The average review length is about 35 words, and the average number of sentences is 3. The dataset contains 937,033 reviews paired with attributes. Specifically, we have 80,256 books, 19,675 users, and 5 rating levels. The word vocabulary size is 161K. Then, the whole dataset is randomly split into TRAIN, DEV, and TEST (70%/10%/20%). The dataset is available at <https://goo.gl/TFjEH4>.

4.2 Settings

We used NLTK (Bird et al., 2009) to tokenize the reviews, and employed the Wikipedia list of common misspellings to correct misspelled words. We kept words that appeared more than 10 times in our vocabulary. The training hyperparameters are selected based on the results of the DEV set. The dimension of attribute vectors is set to 64. The dimensions of word embeddings and hidden vectors are set to 512 in the sequence decoder. Moreover, we stack two layers of recurrent neural networks with LSTM units to generate reviews. All the parameters are randomly initialized by sampling from a uniform distribution $[-0.08, 0.08]$. The batch size, smoothing constant and base learning rate of RMSProp are set to 50, 0.95 and 0.002, respectively. After 10 epochs, the learning rate is decreased by a factor of 0.97 at the end of every epoch as suggested in Karpathy et al. (2016). The dropout rate is set to 0.2 for regularization. We also clamp gradient values into the range $[-5, 5]$ to avoid the exploding gradient problem (Pascanu et al., 2013). The number of epochs is determined by early stopping on the DEV set. At test time,

Method	BLEU-4 (%)	BLEU-1 (%)
Rand	0.86	20.36
MELM	1.28	21.59
NN-pr	1.53	22.44
NN-ur	3.61	26.37
Att2Seq	4.51	30.24
Att2Seq+A	5.03*	30.48*

Table 1: Evaluation results on the TEST set of Amazon data. *: significantly better than the second best score ($p < 0.05$).

we use the greedy search algorithm to generate reviews.

4.3 Evaluation Results

The BLEU (Papineni et al., 2002) score is used for automatic evaluation, which has been shown to correlate well with human judgment on many generation tasks. The BLEU score measures the precision of n-gram matching by comparing the generated results with references, and penalizes length using a brevity penalty term. We compute BLEU-1 (unigram) and BLEU-4 (up to 4 grams) in experiments.

4.3.1 Comparison with Baseline Methods

We describe the comparison methods as follows:

Rand. The predicted results are randomly sampled from all the reviews in the TRAIN set. This baseline method suggests the expected lower bound for this task.

MELM. Maximum Entropy Language Model uses n-gram (up to trigram) features, and the feature template attribute&n-gram (up to bigram). The feature hashing technique is employed to reduce memory usage in each feature group. Noise contrastive estimation (Gutmann and Hyvriinen, 2010) is used to accelerate the training by dropping the normalization term, with 20 contrastive samples in training.

NN-pr. This Nearest Neighbor based method retrieves the reviews that have the same product ID and rating as the input attributes in the TRAIN set. Then we randomly choose a review from them, and use it as the prediction.

NN-ur. The same method as NN-pr but uses both user ID and rating to retrieve candidate reviews.

Att2Seq. Our attribute-to-sequence method described in Section 3. Notice that the attention model is not used.

Method	MELM	Att2Seq	Att2Seq+A
Accuracy (%)	59.00	88.67	93.33*

Table 2: We manually annotate some polarity labels (positive or negative) for generated reviews and compute accuracy by comparing them with the input ratings. *: significantly better than the second best accuracy ($p < 0.05$).

Att2Seq+A. Our method with an attention mechanism.

As shown in Table 1, we compute BLEU scores for these methods. The results of random guess indicate that this task is non-trivial to obtain reasonable performance. MELM performs worse than nearest neighbor search due to the sparsity of lexicalized features, while our model employs distributed representations to avoid using sparse indicator features. Then, we evaluate the NN methods that use different attributes to retrieve reviews, which is a strong baseline for the generation task. The results show that our method outperforms the baseline methods. Moreover, the improvements of the attention mechanism are significant with $p < 0.05$ according to the bootstrap resampling test (KoeHN, 2004). We further show some examples to analyze the attention model in Section 4.4.

4.3.2 Polarity of Generated Reviews

In order to evaluate whether the polarities of generated reviews correspond to their input ratings, we randomly sample some generated reviews and manually annotate their polarity labels. Specifically, we regard the rating 1-2 as negative and 4-5 as positive, and then evaluate performance by computing their classification accuracy. We randomly sample 150 negative examples and 150 positive examples for each method. Next, we ask two graduate students to classify the generated reviews to positive, negative, and indeterminable/neutral. About 93% of examples are annotated with the same labels by two annotators. Table 2 shows our method significantly outperforms others ($p < 0.05$). For Att2Seq+A, some generated reviews are classified to indeterminable/neutral because they contain mixed opinions towards different aspects of books.

4.3.3 Ablation Experiments

In order to evaluate the contributions of model components in our method, we compare to the variants of our model. These models are described

Method	BLEU-4 (%)	BLEU-1 (%)
Att2Seq+A	5.01	30.23
AvgEnc	4.07	28.13
NoStack	4.73	29.58
w/o user	4.10	26.87
w/o product	4.13	27.15
w/o rating	4.12	27.98

Table 3: Model ablation results on the DEV set.

as follows:

AvgEnc. This model uses the average of attribute vectors as the encoding vector, rather than multilayer perceptrons.

NoStack. The method only uses one-layer recurrent neural networks for the sequence decoder.

w/o user/product/rating. This variant does not use the corresponding attribute as input. These results indicate the importance of different information for our model.

As shown in Table 3, we compute BLEU-4 and BLEU-1 scores for our full model and the different variants on the DEV set. The ablation model AvgEnc performs worse than Att2Seq+A. This indicates that multilayer perceptrons can better handle interactions between attributes, outperforming simple averaging of input vectors. Next, we compare to the model without stacking multiple layers of recurrent neural networks as described in Section 3.2. The results demonstrate that deep architectures can improve generation performance. For the next group of variants, we find that removing user, product and rating information harms performance, which indicates that all three attributes contribute to generating relevant reviews.

4.4 The Attention Mechanism

As described in Section 3.3, the attention mechanism learns soft alignment scores between generated words and input attributes. These scores are used to obtain encoder-side context vectors that can better utilize attribute information to predict the next word.

Figure 4 shows three generated examples with different input ratings. The attention scores are represented by gray scales and are column-wisely normalized as described in Equation (7). Firstly, we explain the attention scores over rating information. The input rating of the first example is 1. We find that the phrases “*n’t expecting much*”, “*n’t like*” and “*a little too much*” have larger attention scores on the rating attribute. This demon-

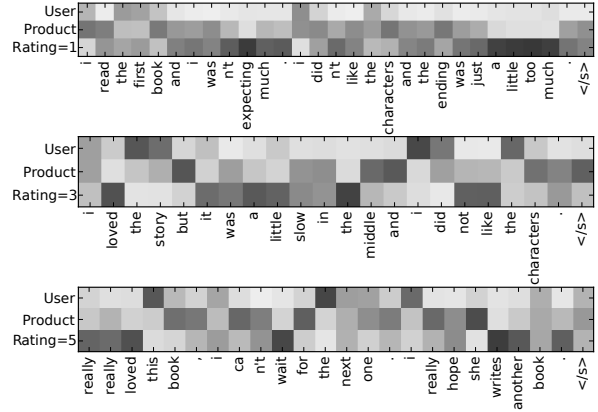


Figure 4: Examples of attention scores (Equation (7)) over three attributes. Darker color indicates higher attention score.

strates rating information has more effect on generating these sentiment words. Next, we increase the rating score to 3 in the second example. The generated review expresses a mixed opinion for different aspects of the book. As indicated by the attention scores, we know that the sentiment words “*loved*”, “*little slow*”, and “*not like*” attend more to rating information. The last example is a positive review with a rating of 5. The attention scores demonstrate the phrases “*loved*”, “*ca n’t wait*”, and “*hope * writes another book*” are used to express polarity. Similarly, the attention scores over user and product information indicate how the generated words are aligned with these two input attributes. For instance, the word “*characters*” has higher attention scores over the product attribute in the first and second example. This indicates that users tend to comment about the characters in this book’s reviews.

4.5 Generated Examples

As shown in Table 4, we sample products and users to generate some examples with different ratings. The special unknown token is removed from the vocabulary of the decoder in the generation process. We keep two attributes fixed and change the other one in every group to show the effects of input.

In the first group, we change the rating from 1 to 5 and keep the others unchanged. The results show that the polarity of generated reviews changes with the rating. For instance, the words “*nice*” and “*liked*” are used for the rating of 3, while the words “*very good*” and “*enjoyed*” are employed for the rating of 5. Moreover, both ex-

U	P	R	Generated Review
A	V	1	i'm sorry to say this was a very boring book. i didn't finish it. i'm not a new fan of the series, but this was a disappointment.
A	V	3	this was a nice story. i liked the characters and the story line. i'm not sure i'd read another by this author.
A	V	5	this was a very good book. i enjoyed the characters and the story line. i'm looking forward to reading more in this series.
B	W	5	i couldn't put it down. it was a great love story. i can't wait to read the next one.
C	W	5	enjoyable story that keeps you turning the pages. the characters are well developed and the plot is excellent. i would recommend this book to anyone who enjoys a good love story.
D	W	5	i loved this book. i could not put it down. i loved this story and the characters. i will be reading the next book.
E	X	1	i read this book because i was looking for something to read. this book was just too much like the others. i thought the author was going to be a good writer, but i was disappointed.
E	Y	1	i was disappointed. i read the first chapter and then i was bored. i read the whole thing, but i just couldn't get into it.
E	Z	1	this book was just too much. i read the whole thing, but i didn't like the way the author ended it. i was hoping for a different ending.

Table 4: **U**: User. **P**: Product. **R**: Rating. This table shows some generated examples of the Att2Seq+A model. In every group, two attributes are kept unchanged, while the other attribute has different values. For instance, in the first group, we use different ratings ranging from 1 (the lowest score) to 5 (the highest score) with the same user and product to generate reviews. The users and products are anonymized by A-E and V-Z.

amples describe “*characters*” and “*story line*”, and are written in the similar styles. This indicates that user and product information determines the content and style of generated reviews, while rating affects the choice of sentiment words. In the next group, we use different user IDs as input attributes. This book is one of the *Fatal Series* written by *Marie Force*, which tells a romantic love story. The first and third examples mention “*next one/book*”, and both the first two reviews contain the phrase “*love story*”. This demonstrates the generated reviews agree with the input product information. In the third group, the attributes, except product ID, are kept unchanged. The examples show our model generates varied reviews for different products.

5 Conclusion

In this paper, we proposed a novel product review generation task, in which generated reviews are conditioned on input attributes. For this task, we formulated a neural network based attribute-to-sequence model that uses multilayer perceptrons to encode input attributes and employs recurrent neural networks to generate reviews. Moreover, we introduced an attention mechanism to better

utilize input attribute information. Additionally, we built a dataset of Amazon product reviews to conduct evaluations. The proposed model consistently outperforms the nearest neighbor search and maximum entropy language model baselines. Besides, the attention mechanism significantly improves the vanilla attribute-to-sequence model. This work suggests several interesting directions for future research. We could use more fine-grained attributes as the input of our model. For example, the generated reviews could be conditioned on device specification, brand, user’s gender, product description, or ratings of a product’s various aspects. Moreover, we could leverage review texts without attributes to improve the sequence decoder.

Acknowledgments

The support of the European Research Council under award number 681760 “Translating Multiple Modalities into Text” is gratefully acknowledged.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly

- learning to align and translate. In *International Conference on Learning Representations*.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.
- J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634.
- Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.
- Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14*, pages 1537–1543. AAAI Press.
- A. Dosovitskiy, J. T. Springenberg, and T. Brox. 2015. Learning to generate chairs with convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1538–1546.
- M. Gutmann and A. Hyvriinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y.W. Teh and M. Titterton, editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR WCP*, pages 297–304. Journal of Machine Learning Research - Proceedings Track.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.
- Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *International Conference on Learning Representations*.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.
- Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.
- Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 369–378, Jeju Island, Korea. Association for Computational Linguistics.
- Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48(1):305–346.
- Zachary C. Lipton, Sharad Vikram, and Julian McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *arXiv preprint arXiv:1511.03683*.
- Bing Liu. 2015. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.
- Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.
- Umar Maqsood. 2015. Synthetic text generation for sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 156–161, Lisboa, Portugal. Association for Computational Linguistics.
- Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15*, pages 785–794, New York, NY, USA. ACM.
- Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

- Dae Hoon Park, Hyun Duk Kim, ChengXiang Zhai, and Lifan Guo. 2015. Retrieval of relevant opinion sentences for new products. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 393–402, New York, NY, USA. ACM.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.
- Richard Socher, Cliff Chung-Yu Lin, Andrew Ng, and Chris Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 129–136, New York, NY, USA. ACM.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.
- T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Technical report.
- Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4534–4542.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015a. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.
- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings.
- Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791.
- Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *International Conference on Learning Representations*.

Learning to generate one-sentence biographies from Wikidata

Andrew Chisholm

University of Sydney
Sydney, Australia

andy.chisholm.89@gmail.com

Will Radford

Hugo Australia
Sydney, Australia

wradford@hugo.ai

Ben Hachey

Hugo Australia
Sydney, Australia

bhachey@hugo.ai

Abstract

We investigate the generation of one-sentence Wikipedia biographies from facts derived from Wikidata slot-value pairs. We train a recurrent neural network sequence-to-sequence model with attention to select facts and generate textual summaries. Our model incorporates a novel secondary objective that helps ensure it generates sentences that contain the input facts. The model achieves a BLEU score of 41, improving significantly upon the vanilla sequence-to-sequence model and scoring roughly twice that of a simple template baseline. Human preference evaluation suggests the model is nearly as good as the Wikipedia reference. Manual analysis explores content selection, suggesting the model can trade the ability to infer knowledge against the risk of hallucinating incorrect information.

1 Introduction

Despite massive effort, Wikipedia and other collaborative knowledge bases (KBs) have coverage and quality problems. Popular topics are covered in great detail, but there is a long tail of specialist topics with little or no text. Other text can be incorrect, whether by accident or vandalism. We report on the task of generating textual summaries for people, mapping slot-value facts to one-sentence encyclopaedic biographies. In addition to initialising stub articles with only structured data, the resulting model could be used to improve consistency and accuracy of existing articles. Figure 1 shows a Wikidata entry for *Mathias Tuomi*, with fact keys and values flattened into a sequence, and the first sentence from his Wikipedia article. Some values are in the text, others are missing

```
TITLE mathias tuomi SEX_OR_GENDER
male DATE_OF_BIRTH 1985-09-03
OCCUPATION squash player
CITIZENSHIP finland
```

Figure 1: Example Wikidata facts encoded as a flat input string. The first sentence of the Wikipedia article reads: *Mathias Tuomi, (born September 30, 1985 in Espoo) is a professional squash player who represents Finland.*

(e.g. *male*) or expressed differently (e.g. *dates*).

We treat this *knowledge-to-text* task like translation, using a recurrent neural network (RNN) sequence-to-sequence model (Sutskever et al., 2014) that learns to select and realise the most salient facts as text. This includes an attention mechanism to focus generation on specific facts, a shared vocabulary over input and output, and a multi-task autoencoding objective for the complementary extraction task. We create a reference dataset comprising more than 400,000 knowledge-text pairs, handling the 15 most frequent slots. We also describe a simple template baseline for comparison on BLEU and crowd-sourced human preference judgements over a heldout TEST set.

Our model obtains a BLEU score of 41.0, compared to 33.1 without the autoencoder and 21.1 for the template baseline. In a crowdsourced preference evaluation, the model outperforms the baseline and is preferred 40% of the time to the Wikipedia reference. Manual analysis of content selection suggests that the model can infer knowledge but also makes mistakes, and that the autoencoding objective encourages the model to select more facts without increasing sentence length. The task formulation and models are a foundation for text completion and consistency in KBs.

2 Background

RNN sequence-to-sequence models (Sutskever et al., 2014) have driven various recent advances in natural language understanding. While initial work focused on problems that were sequences of the same units, such as translating a sequence of words from one language to another, other work has been able to use these models by *coercing* different structures into sequences, e.g., flattening trees for parsing (Vinyals et al., 2015), predicting span types and lengths over byte input (Gillick et al., 2016) or flattening logical forms for semantic parsing (Xiao et al., 2016).

RNNs have also been used successfully in *knowledge-to-text* tasks for human-facing systems, e.g., generating conversational responses (Vinyals and Le, 2015), abstractive summarisation (Rush et al., 2015). Recurrent LSTM models have been used with some success to generate text that completely expresses a set of facts: restaurant recommendation text from dialogue acts (Wen et al., 2015), weather reports from sensor data and sports commentary from on-field events (Mei et al., 2015). Similarly, we learn an end-to-end model trained over key-value facts by flattening them into a sequence.

Choosing the salient and consistent set of facts to include in generated output is also difficult. Recent work explores unsupervised autoencoding objectives in sequence-to-sequence models, improving both text classification as a pretraining step (Dai and Le, 2015) and translation as a multi-task objective (Luong et al., 2016). Our work explores an autoencoding objective which selects content as it generates by constraining the text output sequence to be predictive of the input.

Biographic summarisation has been extensively researched and is often approached as a sequence of subtasks (Schiffman et al., 2001). A version of the task was featured in the Document Understanding Conference in 2004 (Blair-Goldensohn et al., 2004) and other work learns policies for content selection without generating text (Duboue and McKeown, 2003; Zhang et al., 2012; Cheng et al., 2015). While pipeline components can be individually useful, integrating selection and generation allows the model to exploit the interaction between them.

KBs have been used to investigate the interaction between structured facts and unstructured text. Generating textual templates that are filled

by structured data is a common approach and has been used for conversational text (Han et al., 2015) and biographical text generation (Duma and Klein, 2013). Wikipedia has also been a popular resource for studying biography, including sentence harvesting and ordering (Biadys et al., 2008), unsupervised discovery of distinct sequences of life events (Bamman and Smith, 2014) and fact extraction from text (Garera and Yarowsky, 2009). There has also been substantial work in generating from other structured KBs using template induction (Kondadadi et al., 2013), semantic web techniques (Power and Third, 2010), tree adjoining grammars (Gyawali and Gardent, 2014), probabilistic context free grammars (Konstas and Lapata, 2012) and probabilistic models that jointly select and realise content (Angeli et al., 2010).

Lebret et al. (2016) present the closest work to ours with a similar task using Wikipedia infoboxes in place of Wikidata. They condition an attentional neural language model (NLM) on local and global properties of infobox tables, including *copy actions* that allow wholesale insertion of values into generated text. They use 723k sentences from Wikipedia articles with 403k lower-cased words mapping to 1,740 distinct facts. They compare to a 5-gram language-model with copy actions, and find that the NLM has higher BLEU and lower perplexity than their baseline. In contrast, we utilise a deep recurrent model for input encoding, minimal slot value templating and greedy output decoding. We also explore a novel autoencoding objective that measures whether input facts can be re-created from the generated sentence.

Evaluating generated text is challenging and no one metric seems appropriate to measure overall performance. Lebret et al. (2016) report BLEU scores (Papineni et al., 2002) which calculate the n-gram overlap between text produced by the system with respect to a human-written reference. Summarisation evaluations have concentrated on the content that is included in the summary, with semantic content typically extracted manually for comparison (Lin and Hovy, 2003; Nenkova and Passonneau, 2004). We draw from summarisation and generation to formulate a comprehensive evaluation based on automated metrics and human validation. Our final system comparison follows Kondadadi et al. (2013) in running a crowd task to collect pairwise preferences for evaluating and comparing both systems and references.

Fact	Count	%
TITLE (name)	1,011,682	98
SEX_OR_GENDER	1,007,575	0
DATE_OF_BIRTH	817,942	88
OCCUPATION	720,080	67
CITIZENSHIP	663,707	52
DATE_OF_DEATH	346,168	86
PLACE_OF_BIRTH	298,374	25
EDUCATED_AT	141,334	32
SPORTS_TEAM	108,222	29
PLACE_OF_DEATH	107,188	17
POSITION_HELD	87,656	75
PARICIPANT_OF	77,795	23
POLITICAL_PARTY	74,371	49
AWARD_RECEIVED	67,930	44
SPORT	36,950	72

Table 1: The top fifteen slots across entities used for input, and the % of time the value is a substring in the entity’s first sentence.

3 Task and Data

We formulate the one-sentence biography generation task as shown in Figure 1. Input is a flat string representation of the structured data from the KB, comprising slot-value pairs (the subject being the topic of the KB record, e.g., *Mathias Tuomi*), ordered by slot frequency from most to least common. Output is a biography string describing the salient information in one sentence.

We validate the task and evaluation using a closely-aligned set of resources: Wikipedia and Wikidata. In addition to the KB maintenance issues discussed in the introduction, Wikipedia first sentences are of particular interest because they are clear and concise biographical summaries. These could be applied to entities outside Wikipedia for which one can obtain comparable parallel structured/textual data, e.g., movie summaries from IMDb, resume overviews from LinkedIn, product descriptions from Amazon.

We use snapshots of Wikidata (2015/07/13) and Wikipedia (2015/10/02) and batch process them to extract instances for learning. We select all entities that are `INSTANCE_OF human` in Wikidata. We then use `sitelinks` to identify each entity’s Wikipedia article text and NLTK (Bird et al., 2009) to tokenize and extract the lower-cased first sentence. This results in 1,268,515 raw knowledge-text pairs. The summary sentences can be long and the most frequent length is 21 tokens. We filter to

only include those between the 10th and 90th percentiles: 10 and 37 tokens. We split this collection into TRAIN, DEV and TEST collections with 80%, 10% and 10% of instances allocated respectively. Given the large variety of slots which may exist for an entity, we restrict the set of slots used to the top-15 by occurrence frequency. This criteria covers 72.8% of all facts. Table 1 shows the distribution of fact slots in the structured data and the percentage of time tokens from a fact value occur in the corresponding Wikipedia summary.

Additionally, some Wikidata entities remain underpopulated and do not contain sufficient facts to reconstruct a text summary. We control for this information mismatch by limiting our dataset to include only instances with at least 6 facts present. The final dataset includes 401,742 TRAIN, 50,017 DEV and 50,030 TEST instances. Of these instances, 95% contain 6 to 8 slot values while 0.1% contain the maximum of 10 slots. 51% of unique slot-value pairs expressed in TEST and DEV are not observed in TRAIN so generalisation of slot usage is required for the task. The KB facts give us an opportunity to measure the correctness of the generated text in a more precise way than text-to-text tasks. We use this for analysis in Section 7.3, driving insight into system characteristics and implications for use.

3.1 Task complexity

Wikipedia first sentences exhibit a relatively narrow domain of language in comparison to other generation tasks such as translation. As such, it is not clear how complex the generation task is, and we first try to use perplexity to describe this.

We train both RNN models until DEV perplexity stops improving. Our basic sequence-to-sequence model (S2S) reaches perplexity of 2.82 on TRAIN and 2.92 on DEV after 15,000 batches of stochastic gradient descent. The autoencoding sequence-to-sequence model (S2S+AE) takes longer to fit, but reaches a lower minimum perplexity of 2.39 on TRAIN and 2.51 on DEV after 25,000 batches.

To help ground perplexity numbers and understand the complexity of sentence biographies we train a benchmark language model and evaluate perplexity on DEV. Following Lebre et al. (2016), we build Kneser-Ney smoothed 5-gram language models using the KenLM toolkit (Heafield, 2011).

Table 2 lists perplexity numbers for the benchmark LM models with different templating

Templates	DEV
None	29.8
Title	14.5
Full	10.1

Table 2: Language model perplexity across templated datasets.

schemes on DEV. We observe decreasing perplexity for data with greater fact value templating. TITLE indicates templating of entity names only, while FULL indicates templating of all fact values by token index as described in Lebret et al. (2016). This shows that templating is an effective way to reduce the sparsity of a task, and that titles account for a large component of this.

Although Lebret et al. (2016) evaluate on a different dataset, we are able to draw some comparisons given the similarity of our task. On their data, the benchmark LM baseline achieves a similar perplexity of 10.5 to ours when following their templating scheme on our dataset - suggesting both samples are of comparable complexity.

4 Model

We model the task as a sequence-to-sequence learning problem. In this setting, a variable length input sequence of entity facts is encoded by a multi-layer RNN into a fixed-length distributed representation. This input representation is then fed into a separate decoder network which estimates a distribution over tokens as output. During training, parameters for both the encoder and decoder networks are optimized to maximize the likelihood of a summary sequence given an observed fact sequence.

Our setting differs from the translation task in that the input is a sequence representation of structured data rather than natural human language. As described above in Section 3, we map Wikidata facts to a sequence of tokens that serves as input to the model as illustrated at the top of Figure 2. Experiments below demonstrate that this is sufficient for end-to-end learning in the generation task addressed here. To generate summaries, our model must both select relevant content and transform it into a well formed sentence. The decoder network includes an attention mechanism (Vinyals et al., 2015) to help facilitate accurate content selection. This allows the network to focus on different parts of the input sequence during inference.

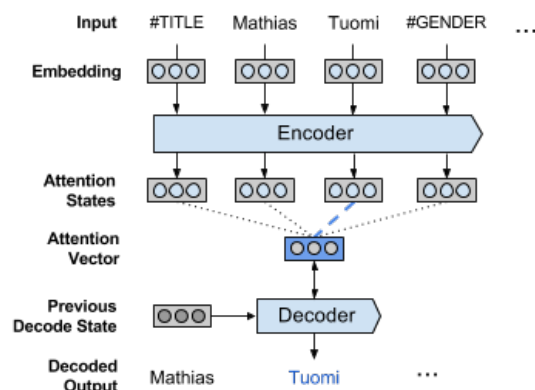


Figure 2: Sequence-to-sequence translation from linearized facts to text.

4.1 Sequence-to-sequence model (s2s)

To generate language, we seed the decoder network with the output of the encoder and a designated GO token. We then generate symbols greedily, taking the most likely output token from the decoder at each step given the preceding sequence until an EOS token is produced. This approach follows (Sutskever et al., 2014) who demonstrate a larger model with greedy sequence inference performs comparably to beam search. In contrast to translation, we might expect good performance on the summarization task where output summary sequences tend to be well structured and often formulaic. Additionally, we expect a partially-shared language across input and output. To exploit this, we use a tied embedding space, which allows both the encoder and decoder networks to share information about word meaning between fact values and output tokens.

Our model uses a 3-layer stacked Gated Recurrent Unit RNN for both encoding and decoding, implemented using TensorFlow.¹ We limit the shared vocabulary to 100,000 tokens with 256 dimensions for each token embedding and hidden layer. Less common tokens are marked as UNK, or unknown. To account for the long tail of entity names, we replace matches of title tokens with templated copy actions (e.g. TITLE0 TITLE1...). These template are then filled after generation, as well as any initial unknown tokens in the output, which we fill with the first title token. We learn using minibatch Stochastic Gradient Descent with a batch size of 64 and a fixed learning rate of 0.5.

¹<https://www.tensorflow.org>, v0.8.

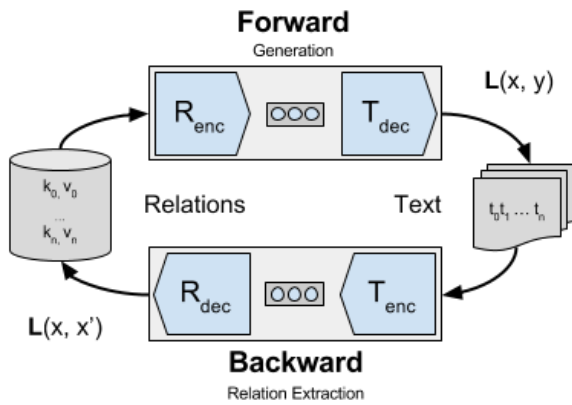


Figure 3: Sequence-to-sequence autoencoder.

4.2 S2S with autoencoding (S2S+AE)

One challenge for vanilla sequence-to-sequence models in this setting is the lack of a mechanism for constraining output sequences to only express those facts present in the data. Given a fact extraction oracle, we might compare facts expressed in the output sequence with those of the input and appropriately adjust the loss for each instance. While a forward-only model is only constrained to generate text sequences predicted by the facts, an autoencoding model is additionally constrained to generate text predictive of the input facts. In place of this ideal setting, we introduce a second sequence-to-sequence model which runs in reverse - re-encoding the text output sequence of the forward model into facts.

This closed-loop model is detailed in Figure 3. The resulting network is trained end-to-end to minimize both the input-to-output sequence loss $L(x, y)$ and output-to-input reconstruction loss $L(x, x')$. While gradients cannot propagate through the greedy forward decode step, shared parameters between the forward and backward network are fit to both tasks. To generate language at test time, the backward network does not need to be evaluated.

5 Experimental methodology

The evaluation suite here includes standard baselines for comparison, automated metrics for learning, human judgement for evaluation and detailed analysis for diagnostics. While each are individually useful, their combination gives a comprehensive analysis of a complex problem space.

5.1 Benchmarks

WIKI We use the first sentence from Wikipedia both as a gold standard reference for evaluating generated sentences, and as an upper bound in human preference evaluation.

BASE Template-based systems are strong baselines, especially in human evaluation. While output may be stilted, the corresponding consistency can be an asset when consistency is important. We induce common patterns from the TRAIN set, replacing full matches of values with their slot and choosing randomly on ties. Multiple non-fact tokens are collapsed to a single symbol. A small sample of the most frequent patterns were manually examined to produce templates, roughly expressed as: TITLE, known as GIVEN_NAME, (born DATE_OF_BIRTH in PLACE_OF_BIRTH; died DATE_OF_DEATH in PLACE_OF_DEATH) is an POSITION_HELD and OCCUPATION from CITIZENSHIP, with some sensible back-offs where slots are not present, and rules for determiner agreement and *is* versus *was* where a death date is present. For example, *ollie freckingham (born 12 november 1988) is a cricketer from the united kingdom.* In total, there are 48 possible template variations.

5.2 Metrics

BLEU We also report BLEU n-gram overlap with respect to the reference Wikipedia summary. With a large dev/test sets (10,000 sentences here), BLEU is a reasonable evaluation of generated content. However, it does not give an indication of well-formedness or readability. Thus we complement BLEU with a human preference evaluation.

Human preference We use crowd-sourced judgements to evaluate the relative quality of generated sentences and the reference Wikipedia first sentence. We obtain pairwise judgements, showing output from two different systems to crowd workers and asking each to give their binary preference. The system name mappings are anonymized and ordered pseudo-randomly. We request 3 judgements and dynamically increase this until we reach at least 70% agreement or a maximum of 5 judgements. We use Crowd-Flower² to collect judgements at the cost of 31 USD for all 6 pairwise combinations over 82

²<http://www.crowdfLOWER.com>

	DEV	TEST
Base	21.3	21.1
S2S	32.5	33.1
S2S+AE	40.5	41.0

Table 3: BLEU scores for each hypothesis against the Wikipedia reference

randomly selected entities. 67 workers contributed judgements to the test data task, each providing no more than 50 responses. We use the majority preference for each comparison. The CrowdFlower agreement is 80.7%, indicating that roughly 4 of 5 votes agree on average.

5.3 Analysis of content selection

Finally, no system is perfect, and it can be challenging to understand the inherent difficulty of the problem space and the limitations of a system. Due to the limitations of the evaluation metrics mentioned above, we propose that manual annotation is important and still required for qualitative analysis to guide system improvement. The structured data in knowledge-to-text tasks allows us, if we can identify expressions of facts in text, cases where facts have been omitted, incorrectly mentioned, or expressed differently.

6 Results

6.1 Comparison against Wikipedia reference

Table 3 shows BLEU scores calculated over 10,000 entities sampled from DEV and TEST using the Wikipedia sentence as a single reference, using uniform weights for 1- to 4-grams, and padding sentences with fewer than 4 tokens. Scores are similar across DEV and TEST, indicating that the samples are of comparable difficulty. We evaluate significance using bootstrapped resampling with 1,000 samples. Each system result lies outside the 95% confidence intervals of other systems. BASE has reasonable scores at 21, with S2S higher at around 32, indicating that the model is at least able to generate closer text than the baseline. S2S+AE scores higher still at around 41, roughly double the baseline scores, indicating that the autoencoder is indeed able to constrain the model to generate better text.

6.2 Human preference evaluation

Table 4 shows the results of our human evaluation over 82 entities sampled from TEST. For each

S2S+AE	BASE	S2S	
60%	61%*	87%**	WIKI
	62%*	77%**	S2S+AE
		65%**	BASE

Table 4: Percentage of entities for which human judges preferred the row system to the column system. E.g., S2S+AE summaries are preferred to BASE for 62% of sample entities.

pair of systems, we show the percentage of entities where the crowd preferred A over B. Significant differences are annotated with * and ** for p values < 0.05 and 0.01 using a one-way χ^2 test. WIKI is uniformly preferred to any system, as is appropriate for an upper bound. The S2S model is the least-preferred with respect to WIKI. The S2S+AE model is more-preferred than the BASE and S2S models, by a larger margin for the latter. These results show that without autoencoding, the sequence-to-sequence model is less effective than a template-based system. Finally, although WIKI is more preferred than S2S+AE, the distributions are not significantly different, which we interpret as evidence that the model is able to generate good text from the human point-of-view, but autoencoding is required to do so.

7 Analysis

While results presented above are encouraging and suggest that the model is performing well, they are not diagnostic in the sense that they can drive deeper insights into model strengths and weaknesses. While inspection and manual analysis is still required, we also leverage the structured factual data inherent to our task to perform quantitative as well as qualitative analysis.

7.1 Fact Count

Figure 4 shows the effects of input fact count on generation performance. While more input facts give more information for the model to work with, longer inputs are also both rarer and more complex to encode. Interestingly, we observe the S2S+AE model maintains performance for more complex inputs while S2S performance declines.

7.2 Example generated text

Table 5 shows some DEV entities and their summaries. The model learns interesting mappings: between numeric and string dates, and country de-

Data		COUNTRY_OF_CITIZENSHIP united states of america DATE_OF_BIRTH 16/04/1927 DATE_OF_DEATH 19/05/1959 OCCUPATION formula one driver PLACE_OF_BIRTH redlands PLACE_OF_DEATH indianapolis SEX_OR_GENDER male TITLE bob cortner
WIKI	n/a	robert charles cortner (april 16 , 1927 may 19 , 1959) was an american automobile racing driver from <i>redlands , california</i> .
BASE	47.7	bob cortner (born 16 april 1927 in redlands ; died 19 may 1959 in indianapolis) was a formula one driver from the united states of america .
S2S	45.7	bob cortner (april 16 , 1927 may 19 , 2005) was an american professional boxer .
S2S+AE	58.8	robert cortner (april 16 , 1927 may 19 , 1959) was an american race-car driver .
Data		COUNTRY_OF_CITIZENSHIP united kingdom DATE_OF_BIRTH 08/01/1906 DATE_OF_DEATH 12/12/1985 OCCUPATION actor PLACE_OF_BIRTH london PLACE_OF_DEATH chelsea SEX_OR_GENDER male TITLE barry mackay (actor)
WIKI	n/a	barry mackay (8 january 1906 12 december 1985) was a british actor .
BASE	34.3	barry mackay (actor) (born 8 january 1906 in london ; died 12 december 1985 in chelsea) was an actor from the united kingdom .
S2S	84.8	barry mackay (8 january 1906 12 december 1985) was a british film actor .
S2S+AE	76.7	barry mackay (8 january 1906 12 december 1985) was an english actor .
Data		COUNTRY_OF_CITIZENSHIP united states of america DATE_OF_BIRTH 27/08/1931 DATE_OF_DEATH 03/11/1995 OCCUPATION jazz musician SEX_OR_GENDER male TITLE joseph "flip" nuñez
WIKI	n/a	joseph " flip ' nuñez was an american jazz pianist , composer , and vocalist of <i>filipino</i> descent .
BASE	15.0	joseph " flip ' nuñez (born 27 august 1931 ; died 3 november 1995) was a jazz musician from the united states of america .
S2S	29.1	joseph " flip ' nuñez (august 27 , 1931 november 3 , 1995) was an american jazz trumpeter .
S2S+AE	29.1	joseph " flip ' nuñez (august 27 , 1931 november 3 , 1995) was an american jazz drummer .

Table 5: Examples of entities from DEV, showing facts, WIKI, BASE, S2S and S2S+AE. We mark **correct**, **incorrect** and *extra* fact values in the text with respect to the Wikidata input.

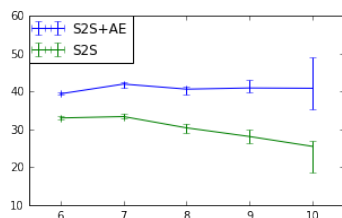


Figure 4: BLEU vs Fact Count on instances from DEV. Error bars indicate the 95% confidence interval for BLEU.

monyms. The model also demonstrates the ability to work around edge cases where templates fail, i.e. stripping parenthetical disambiguations (e.g. (actor)) and emitting the name Robert when the input is Bob. Output also suggests the model may perform inference across multiple facts to improve generation precision, e.g. describing an entity as english rather than british given information about both citizenship and place of birth. Unfortunately, the model can also infer unsubstantiated facts into the text (i.e. jazz drummer).

7.3 Content selection and hallucination

We randomly sample 50 entities from DEV and manually annotate the Wikipedia and system text. We note which fact slots are expressed as well as whether the expressed values are correct with respect to Wikidata. Given two sets of correctly extracted facts, we can consider one *gold*, one *system* and calculate set-based precision, recall and F1.

What percentage of facts are used in the reference summaries? Firstly, to understand how Wikipedia editors select content for the first sentence of articles, we measure recall with the real facts as gold, and Wikipedia as system. Overall, the recall is 0.61 indicating that 61% of input facts are expressed in the reference summary from Wikipedia. The entity name (TITLE) is always expressed. Four slots are nearly always expressed when available: OCCUPATION (90%), DATE_OF_BIRTH (84%), CITIZENSHIP (81%), DATE_OF_DEATH (80%). Six slots are infrequently expressed in the analysis sample: PLACE_OF_BIRTH (33%), POSITION_HELD (25%), PARTICIPANT_OF (20%), POLITICAL_PARTY (20%), EDUCATED_AT (14%), SPORTS_TEAM (9%). Two are never expressed explicitly: PLACE_OF_DEATH (0%), SEX_OR_GENDER (0%). AWARD_RECEIVED and SPORT are not in the analysis sample.

Do systems select the same facts found in the reference summaries? Table 6 shows content selection scores for systems with respect to the Wikipedia text as reference. This suggests that the autoencoding in S2S+AE helps increase fact recall without sacrificing precision. The template baseline also attains this higher recall, but at the cost of precision. For commonly expressed facts found in most person biographies, recall is over 0.95 (e.g., CITIZENSHIP, BIRTH_DATE, DEATH_DATE and OCCUPATION). Facts that are infrequently expressed are more difficult to select, with system F1 ranging from 0.00 to 0.50. Interestingly, macro-averaged F1 across infrequently expressed facts mirror human preference rather than BLEU results, with S2S+AE (0.26) > BASE (0.17) > S2S (0.07). However, all systems perform poorly on these facts and no reliable differences are observed.

How does autoencoding effect fact density? Interestingly, we observe that the autoencoding objective encourages the model to select more

	P	R	F
BASE	0.80	0.79	0.79
S2S	0.89	0.67	0.77
S2S+AE	0.89	0.78	0.83

Table 6: Fact-set content selection results phrased as precision, recall and F1 of systems with respect to the Wikipedia reference on DEV.

	P	R	F
BASE	1.00	0.74	0.85
S2S	0.96	0.55	0.70
S2S+AE	0.93	0.62	0.74
WIKI	0.81	0.61	0.69

Table 7: Hallucination results phrased as precision, recall and F1 of systems with respect to the Wikidata input on DEV.

facts (5.2 for S2S+AE vs. 4.5 for S2S), without increasing sentence length (19.1 vs. 19.7 tokens). BASE is similarly productive (5.1 facts) but wordier (21.2 tokens), while the WIKI reference produces both more facts (6.1) and longer sentences (23.7).

Do systems hallucinate facts? To quantify the effect of hallucinated facts, we assess content selection scores of systems with respect to the input Wikidata relations (Table 7). Our best model achieves a precision of 0.93 with respect to Wikidata input. Notably, the template-driven baseline maintains a precision of 1.0 as it is constrained to emit Wikidata facts verbatim.

8 Discussion and future work

Our experiments show that RNNs can generate biographic summaries from structured data, and that a secondary autoencoding objective is able to account for some of the information mismatch between input facts and target output sentences. In the future, we will explore whether results improve with explicit modelling of facts and conditioning of generation and autoencoding losses on slots. We expect this could benefit generation for diverse and noisy slot schemas like Wikipedia Infoboxes.

Another natural extension is to investigate the performance of the network running in reverse, from summary text back to facts. We plan to isolate the performance of the S2S+AE backward model when inferring facts and compare it to stan-

standard relation extraction systems. Finally, similar RNN models have been applied extensively to language translation tasks. We plan to explore whether a joint model of machine translation and fact-driven generation can help populate KB entries for low-coverage languages by leveraging a shared set of facts.

9 Conclusion

We present a neural model for mapping between structured and unstructured data, focusing on creating Wikipedia biographic summary sentences from Wikidata slot-value pairs. We introduce a sequence-to-sequence autoencoding RNN which improves upon base models by jointly learning to generate text and reconstruct facts. Our analysis of the task suggests evaluation in this domain is challenging. In place of a single score, we analyse statistical measures, human preference judgments and manual annotation to help characterise the task and understand system performance. In the human preference evaluation, our best model outperforms template baselines and is preferred 40% of the time to the gold standard Wikipedia reference.

Code and data is available at <https://github.com/andychisholm/mimo>.

Acknowledgments

This work was supported by a Google Faculty Research Award (Chisholm) and an Australian Research Council Discovery Early Career Researcher Award (DE120102900, Hachey). Many thanks to reviewers for insightful comments and suggestions, and to Glen Pink, Kellie Webster, Art Harol and Bo Han for feedback at various stages.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Conference on Empirical Methods in Natural Language Processing*, pages 502–512.

David Bamman and Noah A. Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics*, 2:363–376.

Fadi Biadsy, Julia Hirschberg, and Elena Filatova. 2008. An unsupervised approach to biography production using Wikipedia. In *Annual Meeting of the Association for Computational Linguistics*, pages 807–815.

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

Sasha Blair-Goldensohn, David Evans, Vasileios Hatzivassiloglou, Kathleen McKeown, Ani Nenkova, Rebecca Passonneau, Barry Schiffman, Andrew Schlaikjer, Advaith Siddharthan, and Sergey Siegelman. 2004. Columbia University at DUC 2004. In *Proceedings of the Document Understanding Workshop*, pages 23–30.

Gong Cheng, Danyun Xu, and Yuzhong Qu. 2015. Summarizing entity descriptions for effective and efficient human-centered entity linking. In *International Conference on World Wide Web*, pages 184–194.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Annual Conference on Neural Information Processing Systems*, pages 3079–3087.

Pablo Ariel Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Conference on Empirical Methods in Natural Language Processing*, pages 121–128.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *International Conference on Computational Semantics*, pages 83–94.

Nikesh Garera and David Yarowsky. 2009. Structural, transitive and latent models for biographic fact extraction. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 300–308.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1296–1306.

Bikash Gyawali and Claire Gardent. 2014. Surface realisation from knowledge-bases. In *Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. In *Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 129–133.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Workshop on Statistical Machine Translation*, pages 187–197.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical NLG framework for aggregated planning and realization. In *Annual Meeting of the*

- Association for Computational Linguistics*, pages 1406–1415.
- Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Annual Meeting of the Association for Computational Linguistics*, pages 369–378.
- Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.
- Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 71–78.
- Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *International Conference on Learning Representations*.
- Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2015. What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 720–730.
- Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 145–152.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Annual Meeting on Association for Computational Linguistics*, pages 311–318.
- Richard Power and Allan Third. 2010. Expressing OWL axioms by english sentences: Dubious in theory, feasible in practice. In *International Conference on Computational Linguistics*, pages 1006–1013.
- Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Conference on Empirical Methods in Natural Language Processing*, pages 379–389.
- Barry Schiffman, Inderjeet Mani, and Kristian Conception. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In *Annual Meeting of the Association for Computational Linguistics*, pages 458–465.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Annual Conference on Neural Information Processing Systems*, pages 3104–3112.
- Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.
- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Annual Conference on Neural Information Processing Systems*, pages 2755–2763.
- Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.
- Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Annual Meeting of the Association for Computational Linguistics*, pages 1341–1350.
- Lanbo Zhang, Yi Zhang, and Yunfei Chen. 2012. Summarizing highly structured documents for effective search interaction. In *International Conference on Research and Development in Information Retrieval*, pages 145–154.

Transition-Based Deep Input Linearization

Ratish Puduppully ^{†*}, Yue Zhang [‡] and Manish Shrivastava [†]

[†]Kohli Center on Intelligent Systems (KCIS),

International Institute of Information Technology, Hyderabad (IIIT Hyderabad)

[‡]Singapore University of Technology and Design

ratish.surendran@research.iiit.ac.in, yue_zhang@sutd.edu.sg,

m.shrivastava@iiit.ac.in

Abstract

Traditional methods for deep NLG adopt pipeline approaches comprising stages such as constructing syntactic input, predicting function words, linearizing the syntactic input and generating the surface forms. Though easier to visualize, pipeline approaches suffer from error propagation. In addition, information available across modules cannot be leveraged by all modules. We construct a transition-based model to jointly perform linearization, function word prediction and morphological generation, which considerably improves upon the accuracy compared to a pipelined baseline system. On a standard deep input linearization shared task, our system achieves the best results reported so far.

1 Introduction

Natural language generation (NLG) (Reiter and Dale, 1997; White, 2004) aims to synthesize natural language text given input syntactic, semantic or logical representations. It has been shown useful in various tasks in NLP, including machine translation (Chang and Toutanova, 2007; Zhang et al., 2014), abstractive summarization (Barzilay and McKeown, 2005) and grammatical error correction (Lee and Seneff, 2006).

A line of traditional methods treat the problem as a pipeline of several independent steps (Bohnet et al., 2010; Wan et al., 2009; Bangalore et al., 2000; H. Oh and I. Rudnicky, 2000; Langkilde and Knight, 1998). For example, shown in Figure 1b, a pipeline based on the meaning text theory (MTT) (Melčuk, 1988) splits NLG into three

*Part of the work was done when the author was a visiting student at Singapore University of Technology and Design.

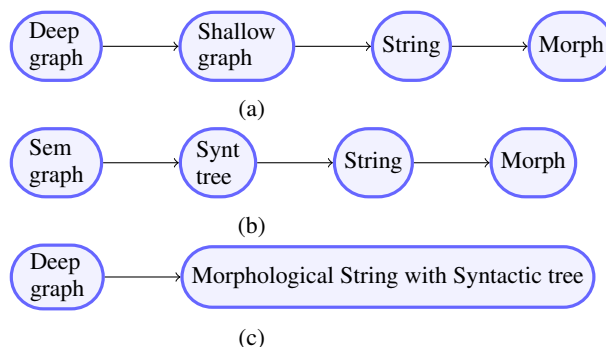


Figure 1: Linearization pipelines (a) NLG pipeline with deep input graph, (b) pipeline based on the meaning text theory, (c) this paper.

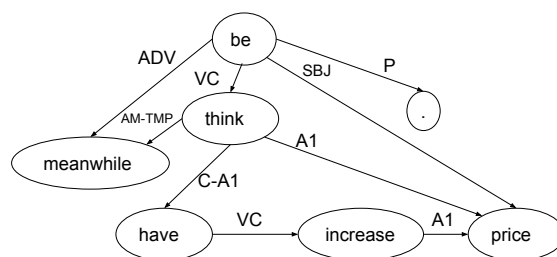


Figure 2: Sample deep graph for the sentence: *meanwhile, prices are thought to have increased.* Note that words are replaced by their lemmas. The function word *to* and comma are absent in graph.

independent steps 1. syntactic generation: generating an unordered and lemma-formed syntactic tree from a semantic graph, introducing function words; 2. syntactic linearization: linearizing the unordered syntactic tree; 3. morphological generation: generating the inflection for each lemma in the string.

In this paper we focus on deep graph as input. Exemplified in Figure 2, the deep input type is intended to be an abstract representation of the meaning of a sentence. Unlike semantic input, where the nodes are semantic representations of input, deep input is more surface centric, with lem-

mas for each word being connected by semantic labels (Banarescu et al., 2013; Melčuk, 2015). In contrast to shallow syntactic trees, function words in surface forms are not included in deep graphs (Belz et al., 2011). Deep inputs can more commonly occur as input of NLG systems where entities and content words are available, and one has to generate a grammatical sentence using them with only provision for inflections of words and introduction of function words. Such usecases include summarization, dialog generation etc.

A pipeline of deep input linearization is shown in Figure 1a. Generation involves predicting the correct word order, deciding inflections and also filling in function words at the appropriate positions. The worst-case complexity is $n!$ for permuting n words, 2^n for function word prediction (assuming that a function word can be inserted after each content word), and 2^n for inflection generation (assuming two morphological forms for each lemma). On the dataset from the First Surface Realisation Shared Task, Bohnet et al. (2011) achieved the best reported results on linearizing deep input representation, following the pipeline of Figure 1b (with input as deep graph instead of semantic graph). They construct a syntactic tree from deep input graph followed by function word prediction, linearization and morphological generation. A rich set of features are used at each stage of the pipeline and for each adjacent pair of stages, an SVM decoder is defined.

Pipelined systems suffer from the problem of error propagation. In addition, because the steps are independent of each other, information available in a later stage is not made use of in the earlier stages. We introduce a transition-based (Nivre, 2008) method for *joint* deep input surface realisation integrating linearization, function word prediction and morphological generation. The model is shown in Fig 1c, as compared with the pipelined baseline in Fig 1a. For a directly comparable baseline, we construct a pipeline system of function words prediction, linearization and morphological generation similar to the pipeline of Bohnet et al. (2011), but with the following difference. Our baseline pipeline system makes function word prediction for a deep input graph, whereas Bohnet et al. (2011) have a preprocessing step to construct a syntactic tree from the deep input graph, which is given as input to the function word prediction module. Our pipeline is directly comparable to the

joint system with regard to the use of information.

Standard evaluations show that: 1. Our joint model for deep input surface realisation achieves significantly better scores over its pipeline counterpart. 2. We achieve the best results reported on the task. Our system scores 1 BLEU point better over Bohnet et al. (2011) without using any external resources. We make the source code available at <https://github.com/SUTDNLP/ZGen/releases/tag/v0.3>.

2 Related Work

Related work can be broadly summarized into three areas: abstract word ordering, applications of meaning-text theory and joint modelling of NLP tasks. In abstract word ordering (Wan et al., 2009; Zhang, 2013; Zhang and Clark, 2015), De Gispert et al. (2014) compose phrases over individual words and permute the phrases to achieve linearization. Schmaltz et al. (2016) show that strong surface-level language models are more effective than models trained with syntactic information for the task of linearization. Transition-based techniques have also been explored (Liu et al., 2015; Liu and Zhang, 2015; Puduppully et al., 2016). To our knowledge, we are the first to use transition-based techniques for *deep* input linearization.

There has been work done in the area of sentence linearization using meaning-text theory (Melčuk, 1988). Belz et al. (2011) organized a shared task on both shallow and deep linearization according to meaning-text theory, which provides a standard benchmark for system comparison. Song et al. (2014) achieved the best results for the task of *shallow*-syntactic linearization. Using SVM models with rich features, Bohnet et al. (2011) achieved state-of-art results on the task of *deep* realization. While they built a pipeline system, we show that joint models can be used to overcome limitations of the pipeline approach giving the best results.

Joint models for NLP have shown effectiveness in recent years. Though having to tackle increased search space, they overcome issues with error propagation in pipelined models. Joint models have been explored for grammar-based approaches to surface realisation using HPSG and CCG (Carroll and Oepen, 2005; Velldal and Oepen, 2006; Espinosa et al., 2008; White and Rajkumar, 2009; White, 2006; Carroll et al., 1999).

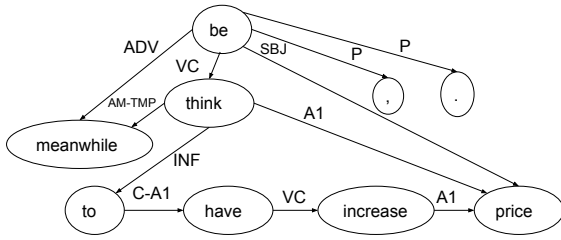


Figure 3: Equivalent shallow graph for Figure 2.

Joint models have been proposed for word segmentation and POS-tagging (Zhang and Clark, 2010), POS-tagging and syntactic chunking (Sutton et al., 2007), segmentation and normalization (Qian et al., 2015), syntactic linearization and morphologization (Song et al., 2014), parsing and NER (Finkel and Manning, 2009), entity and relation extraction (Li and Ji, 2014) and so on. We propose a first joint model for deep realization, integrating linearization, function word prediction and morphological generation.

3 Baseline

We build a baseline following the pipeline in Figure 1a. Three stages are involved: 1. prediction of function words, inserting the predicted function words in the deep graph, resulting in a *shallow* graph; 2. linearizing the shallow graph; 3. generating the inflection for each lemma in the string.

3.1 Function Word Prediction

In the First Surface Realisation Shared Task dataset (Belz et al., 2011), there are three classes of function words to predict: *to* infinitive, *that* complementizer and *comma*. We implement classifiers to predict these classes of function words locally at respective positions in the deep graph resulting in a shallow graph (Figure 3). At each location the input is a node and output is a class indicating if *to* or *that* need to be inserted under the node or the count of *comma* to be introduced under the node.

Table 1 shows the feature templates for classification of *to* infinitives and *that* complementizers and Table 2 shows the feature templates for predicting the count of *comma* child nodes for each non-leaf node in the graph. These feature templates are a subset of features used in the joint model (Section 4) with the exceptions being word order features, which are not available here for the pipeline system, since earlier stages cannot leverage features in subsequent outcomes. We use av-

Features for predicting function words including <i>to</i> infinitive, <i>that</i> complementizer
WORD(n); POS(n); WORD(c)

Table 1: Feature templates for the prediction of function words- *to* infinitive and *that* complementizer. Indices on the surface string: n – word index; c – child of n ; Functions: WORD – word at index; POS – part-of-speech at index.

Features for predicting count of <i>comma</i>
WORD(n); POS(n)
BAG(WORD-MOD(n))
BAG(LABEL-MOD(n))

Table 2: Feature templates for the comma prediction system. Indices on the surface string: n – word index; Functions: WORD – word at index; POS – part-of-speech at index; WORD-MOD – modifiers of index; LABEL-MOD – dependency labels of modifiers; BAG – set.

eraged perceptron classifier (Collins, 2002) to predict function words, which is consistent with the joint model.

3.2 Linearization

The next step is linearizing the graph, which we solve using a novel transition-based algorithm.

3.2.1 Transition-Based Tree Linearization

Liu et al. (2015) introduce a transition-based model for tree linearization. The approach extends from transition-based parsers (Nivre and Scholz, 2004; Chen and Manning, 2014), where *state* consists of *stack* to hold partially built outputs and a *queue* to hold input sequence of words. In case of linearization, the input is a set of words. Liu et al. therefore use a set to hold the input instead of a queue. State is represented by a tuple (σ, ρ, A) , where σ is stack to store partial derivations, ρ is set of input words and A is the set of dependency relations that have been built. There are three transition actions:

- SHIFT-Word-POS – shifts *Word* from ρ , assigns POS to it and pushes it to top of stack as S_0 ;
- LEFTARC-LABEL – constructs dependency arc $S_1 \xleftarrow{LABEL} S_0$ and pops out second element from top of stack S_1 ;
- RIGHTARC-LABEL – constructs dependency arc $S_1 \xrightarrow{LABEL} S_0$ and pops out top of stack S_0 .

Input lemmas: {think₁, price₂, .₃, increase₄, be₅, have₆, meanwhile₇, .₈, to₉}

Transition	σ	ρ	A
0		{1...7}	\emptyset
1	SH-meanwhile	{7}	{1...6,8,9}
2	SH-	{7 8}	{1...6,9}
3	SH-price	{7 8 2}	{1,3,4,5,6,9}
4	SH-be	{7 8 2 5}	{1,3,4,6,9}
5	SH-think	{7 8 2 5 1}	{3,4,6,9}
6	SH-to	{7 8 2 5 1 9}	{3,4,6}
7	SH-have	{7 8 2 5 1 9 6}	{3,4}
8	SH-increase	{7 8 2 5 1 9 6 4}	{3}
9	RA	{7 8 2 5 1 9 6}	{3}
10	RA	{7 8 2 5 1 9}	{3}
11	RA	{7 8 2 5 1}	{3}
12	RA	{7 8 2 5}	{3}
13	SH-	{7 8 2 5 3}	{}
14	RA	{7 8 2 5}	{}
15	LA	{7 8 5}	{}
16	LA	{7 5}	{}
17	LA	{5}	{}

Table 3: Transition action sequence for linearizing the graph in Figure 3. SH - SHIFT, RA - RIGHTARC, LA - LEFTARC. POS is not shown in SHIFT actions.

The sequence of actions to linearize the set {*he*, *goes*, *home*} is SHIFT-*he*, SHIFT-*goes*, SHIFT-*home*, RIGHTARC-OBJ, LEFTARC-SBJ.

The full set of feature templates are shown in Table 2 of Liu et al. (2015), partly shown in Table 4. The features include word(w), POS(p) and dependency label (l) of elements on stack and their descendants S_0 , S_1 , $S_{0,l}$, $S_{0,r}$ etc. For example, word on top of stack is S_0w and word on first left child of S_0 is $S_{0,l}w$. These are called configuration features. They are combined with all possible actions to score the action. Puduppully et al. (2016) extend Liu et al. (2015) by redefining features to address feature sparsity and introduce lookahead features, thereby achieving highest accuracies on task of abstract word ordering.

3.2.2 Shallow Graph Linearization

Our transition based graph linearization system extends from Puduppully et al. (2016). In our case, the input is a shallow graph instead of a syntactic tree, and hence the search space is larger. On the other hand, the same set of actions can still be applied, with additional constraints on valid actions given each configuration (Section 3.2.3). Table 3 shows the sequence of transition actions to linearize shallow graph in Figure 3.

3.2.3 Obtaining Possible Transition Actions Given a Configuration

The purpose of a GETPOSSIBLEACTIONS function is to find out the set of transition actions that can lead to a valid output given a certain state.

Algorithm 1: GETPOSSIBLEACTIONS for shallow graph linearization

Input: A state $s = ([\sigma|j\ i], \rho, A)$ and input graph C
Output: A set of possible transition actions T

```

1  $T \leftarrow \emptyset$ 
2 if  $s.\sigma == \emptyset$  then
3   for  $k \in s.\rho$  do
4      $T \leftarrow T \cup (\text{SHIFT}, \text{POS}, k)$ 
5 else
6   if  $\exists k, k \in (\text{DIRECTCHILDREN}(i) \cap s.\rho)$  then
7      $\text{SHIFTSUBTREE}(i, \rho)$ 
8   else
9     if  $A.\text{LEFTCHILD}(i)$  is NIL then
10       $\text{SHIFTSUBTREE}(i, \rho)$ 
11     if  $\{j \rightarrow i\} \in C \wedge A.\text{LEFTCHILD}(j)$  is NIL then
12       then
13          $T \leftarrow T \cup (\text{RIGHTARC})$ 
14         if  $i \in \text{DESCENDANT}(j)$  then
15            $\text{PROCESSDESCENDANT}(i, j)$ 
16         if  $i \in \text{SIBLING}(j)$  then
17            $\text{PROCESSSIBLING}(i, j)$ 
18       else if  $\{j \leftarrow i\} \in C$  then
19          $T \leftarrow T \cup (\text{LEFTARC})$ 
20         if  $i \in \text{SIBLING}(j)$  then
21            $\text{PROCESSSIBLING}(i, j)$ 
22       else
23         if  $\text{size}(s.\sigma) == 1$  then
24            $\text{SHIFTPARENTANDSIBLINGS}(i)$ 
25         else
26           if  $i \in \text{DESCENDANT}(j)$  then
27              $\text{PROCESSDESCENDANT}(i, j)$ 
28           if  $i \in \text{SIBLING}(j)$  then
29              $\text{PROCESSSIBLING}(i, j)$ 
29 return  $T$ 

```

Algorithm 2: DIRECTCHILDREN

Input: A state $s = ([\sigma|j\ i], \rho, A)$, input_node and graph C .

Output: DC direct child nodes of input node

```

1  $DC \leftarrow \emptyset$ 
2 for  $k \in (C.\text{CHILDREN}(\text{input\_node}))$  do
3    $\text{Parents} \leftarrow C.\text{PARENTS}(k)$ 
4   if  $\text{Parents.size} == 1$  then
5      $DC \leftarrow DC \cup k$ 
6   else
7     for  $m \in \text{Parents}$  do
8       if  $A.\text{LEFTCHILD}(m)$  is not NIL  $\vee m == \text{input\_node}$  then
9         continue
10      if  $m \cap s.\rho$  then
11        goto OutsideLoop
12      if  $m \in \sigma \wedge \sigma.\text{ISANCESTOR}(m, C)$  then
13        goto OutsideLoop
14       $DC \leftarrow DC \cup k$ 
15   OutsideLoop:
16 return  $DC$ 

```

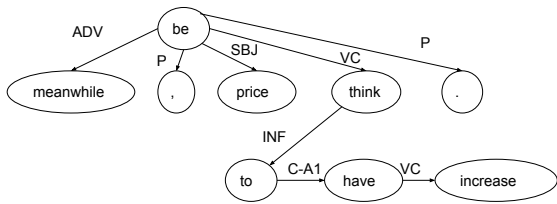


Figure 4: Equivalent syntactic tree for Figure 2.

Algorithm 3: SHIFTSUBTREE

Input: A state $s = ([\sigma|j\ i], \rho, A)$, graph C , head k

Output: a set of possible Transition actions T

```

1  $T \leftarrow \emptyset$ 
2  $T \leftarrow T \cup (\text{SHIFT}, \text{POS}, k)$ 
3  $queue\ q$ 
4  $q.push(k)$ 
5 while  $q$  is not empty do
6    $front = q.pop()$ 
7   for  $m \in (C.CHILDREN(front) \cap s.\rho)$  do
8      $q.push(m)$ 
9      $T \leftarrow T \cup (\text{SHIFT}, \text{POS}, m)$ 

```

This is because not all sequences of actions correspond to a well-formed output. Essentially, given a state $s = ([\sigma|j\ i], \rho, A)$ and an input graph C , the Decoder extracts syntactic tree from the graph (cf. Figure 4 extracted from Figure 3), outputting RIGHTARC, LEFTARC only if the corresponding arc exists in C . The corresponding pseudocode is shown in Algorithm 1.

In particular, if node i has *direct child nodes* in C , the descendants of i are shifted (line 6-7) (see Algorithm 3). Here *direct child nodes* (see Algorithm 2) include those child nodes of i for which i is the only parent or if there is more than one parent then every other parent is shifted on to the stack without possibility to reduce the child node. If no *direct child node* is in the buffer, then all graph descendants of i are shifted. Now, there are three configurations possible between i and j : 1. i and j are directly connected in C . This results in RIGHTARC or LEFTARC action; 2. i is descendant of j . In this case the parents of i (such that they are descendants of j) and siblings of i through such parents are shifted. 3. i is sibling of j . In this case, parents of i and their descendants are shifted such that A remains consistent. Because the input is a graph, more than one of the above configuration can occur simultaneously. More detailed discussion related to GETPOSSIBLEACTIONS is given in Appendix A.

Unigrams
$S_0w; S_0p; S_{0,l}w; S_{0,l}p; S_{0,l}l; S_{0,r}w; S_{0,r}p; S_{0,r}l;$
Bigram
$S_0wS_{0,l}w; S_0wS_{0,l}p; S_0wS_{0,l}l; S_0pS_{0,l}w;$
Linearization
$w_0; p_0; w_{-1}w_0; p_{-1}p_0; w_{-2}w_{-1}w_0; p_{-2}p_{-1}p_0$

Table 4: Baseline linearization feature templates. A subset is shown here. For the full feature set, refer to Table 2 of Liu et al. (2015).

3.2.4 Feature Templates

There are three sets of features. The first is the set of baseline linearization feature templates from Table 2 in Liu et al. (2015), partly shown in Table 4. The second is a set of *lookahead features* similar to that of Puduppully et al. (2016), shown in Table 5.¹ Parent lookahead feature in Puduppully et al. (2016) is defined for the only parent. For graph linearization, however, the parent lookahead feature need to be defined for set of parents. The third set of features in Table 6 are newly introduced for Graph Linearization. Arc_{left} is a binary feature indicating if there is left arc between S_0 and S_1 , whereas Arc_{right} indicates if there is a right arc. $L_{is_descendant}$ is a binary feature indicating if L is descendant of S_0 , and $L_{is_parent_or_sibling}$ indicates if it is a parent or sibling of S_0 . $S_{0descendants_shifted}$ is binary feature indicating if all the descendants of S_0 are shifted.

Not having POS in the input dataset, we compute the feature templates for POS making use of the most frequent POS of the lemma in the gold training data. For the features with dependency labels, we use the input graph labels.

3.2.5 Search and Learning

We follow Puduppully et al. (2016) and Liu et al. (2015), applying the learning and search framework of Zhang and Clark (2011). Pseudocode is shown in Algorithm 4. It performs beam search holding k best states in an agenda at each incremental step. At the start of decoding, agenda holds the initial state. At a step, for each state in the

¹Here L_{cls} represents set of arc labels of child nodes (of word to shift L) shifted on the stack, L_{clns} represents set of arc labels of child nodes not shifted on the stack, L_{cps} the POS set of shifted child nodes, L_{cpns} the POS set of unshifted child nodes, L_{sls} the set of arc labels of shifted siblings, L_{slns} the set of arc labels of unshifted siblings, L_{sps} the POS set of shifted siblings, L_{cpns} the POS set of unshifted siblings, L_{pls} the set of arc labels of shifted parents, L_{plns} the set of arc labels of unshifted parents, L_{pps} the POS set of shifted parents, L_{ppns} the POS set of unshifted parents.

set of label and POS of child nodes of L
$L_{cls}; L_{clns}; L_{cps}; L_{cpns};$ $S_0wL_{cls}; S_0pL_{cls}; S_1wL_{cls}; S_1pL_{cls};$
set of label and POS of first-level siblings of L
$L_{sls}; L_{slns}; L_{sps}; L_{spns};$ $S_0wL_{sls}; S_0pL_{sls}; S_1wL_{sls}; S_1pL_{sls};$
set of label and POS of parents of L
$L_{pls}; L_{plns}; L_{pps}; L_{ppns};$ $S_0wL_{pls}; S_0pL_{pls}; S_1wL_{pls}; S_1pL_{pls};$

Table 5: Lookahead linearization feature templates for the word L to shift. A subset is shown here. For the full feature set, refer to Table 2 of Puduppully et al. (2016). An identical set of feature templates are defined for S_0 .

arc features between S_0 and S_1
$Arc_{left}; Arc_{right};$
lookahead features for L
$L_{is_descendant}; L_{is_parent_or_sibling};$
are all descendants of S_0 shifted
$S_0descendants_shifted;$
feature combination
$S_0descendants_shiftedArc_{left};$ $S_0descendants_shiftedArc_{right};$ $S_0descendants_shifted L_{is_descendant};$ $S_0descendants_shifted L_{is_parent_or_sibling};$

Table 6: Graph linearization feature templates

agenda, each of transition actions in GETPOSSIBLEACTIONS is applied. The top- k states are updated in the agenda for the next step. The process repeats for $2n$ steps as each word needs to be shifted once on to the stack and reduced once. After $2n$ steps, the highest scoring state in agenda is taken as the output. The complexity of algorithm is n^2 , as it takes $2n$ steps to complete and during each step, the number of transition actions is proportional to ρ . Given a configuration C , the score of a possible action a is calculated as:

$$Score(a) = \vec{\theta} \cdot \Phi(\vec{C}, a),$$

where $\vec{\theta}$ is the model parameter vector and $\Phi(\vec{C}, a)$ denotes a feature vector consisting of *configuration* and *action* components. Given a set of labeled training examples, the averaged perceptron with early update (Collins and Roark, 2004) is used.

3.3 Morphological Generation

The last step is to inflate the lemmas in the sentence. There are three POS categories, including nouns, verbs and articles, for which we need to generate morphological forms. We use Wiktionary² as a basis and write a small set of rules

²<https://en.wiktionary.org/>

Algorithm 4: transition-based linearization

Input: C , a set of input syntactic constraints
Output: The highest-scored final state

```

1 candidates  $\leftarrow ([], set(1..n), \emptyset)$ 
2 agenda  $\leftarrow \emptyset$ 
3 for  $i \leftarrow 1..2n$  do
4   for  $s$  in candidates do
5     for action in GETPOSSIBLEACTIONS( $s, C$ ) do
6       agenda  $\leftarrow$  APPLY( $s, action$ )
7   candidates  $\leftarrow$  TOP-K(agenda)
8   agenda  $\leftarrow \emptyset$ 
9 best  $\leftarrow$  BEST(candidates)
10 return best

```

Rules for *be*

```

attr['partic'] == 'pres'  $\rightarrow$  being
attr['partic'] == 'past'  $\rightarrow$  been
attr['tense'] == 'past'
  sbj.attr['num'] == 'sg'  $\rightarrow$  was
  sbj.attr['num'] == 'pl'  $\rightarrow$  were
  other  $\rightarrow$  [was,were]
attr['tense'] == 'pres'
  sbj.attr['num'] == 'sg'  $\rightarrow$  is
  sbj.attr['num'] == 'pl'  $\rightarrow$  are
  other  $\rightarrow$  [am,is,are]

```

Rules for other verbs

```

attr['partic'] == 'pres'  $\rightarrow$  wik.get(lemma, VBG)
attr['partic'] == 'past'  $\rightarrow$  wik.get(lemma, VBN )
attr['tense'] == 'past'  $\rightarrow$  wik.get(lemma, VBD)
attr['tense'] == 'pres'
  sbj.attr['num'] == 'sg'  $\rightarrow$  wik.get(lemma, VBZ )
  other  $\rightarrow$  wik.getall(lemma)

```

Rules for other types

```

lemma==a  $\rightarrow$  [a,an]
lemma==not  $\rightarrow$  [not,n't]
attr['num'] == 'sg'  $\rightarrow$  wik.get(lemma,NNP/NN)
attr['num'] == 'pl'  $\rightarrow$  wik.get(lemma,NNPS/NNS)

```

Table 7: Lemma rules. All rules are in the format: conditions \rightarrow candidate inflections. Nested conditions are listed in multi-lines with indentation. *wik* denotes english wiktionary.

similar to that used in Song et al. (2014), listed in Table 7, to generate a candidate set of inflections. An averaged perceptron classifier (Collins, 2002) is trained for each lemma. For distinguishing between singular and plural candidate verb forms, the feature templates in Table 8 are used.

4 Joint Method

We design a joint method for function word prediction (Section 3.1), linearization (Section 3.2) and morphological generation (Section 3.3) by further extending the transition-based system of Section 3.2, integrating actions for function word prediction and morphological generation.

Features for predicting singular/ plural verb forms
WORD($n-1$)WORD($n-2$)WORD($n-3$); COUNT.SUBJ(n);
COUNT($n-1$)COUNT($n-2$)COUNT($n-3$); SUBJ(n);
WORD($n-1$)WORD($n-2$); COUNT($n-1$)COUNT($n-2$);
WORD($n-1$); COUNT($n-1$); WORD($n+1$); COUNT($n+1$);

Table 8: Feature templates for predicting singular/ plural verb forms. Indices on the surface string: n – word index; Functions: WORD – word at index n ; COUNT – word at n is singular or plural form; SUBJ – word at subject of n ; COUNT.SUBJ – word at subject of n is singular or plural form.

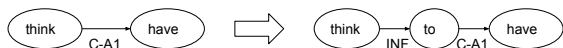


Figure 5: Example for SPLITARC-*to*.

4.1 Transition Actions

In addition to SHIFT, LEFTARC and RIGHTARC in Section 3.2.1, we use the following new transition actions for inserting function words:

- INSERT, inserts comma at the present position;
- SPLITARC-*Word*, splits an arc in the input graph C , inserting a function word between the words connected by the arc. Here *Word* specifies the function word being inserted (Figure 5).

We generate a candidate set of inflections for each lemma following the approach in Section 3.3. For each candidate inflection, we generate a corresponding SHIFT transition action. The rules in Table 7 are used to prune impossible inflections.³

Table 9 shows the transition actions to linearize the graph in Figure 2. These newly introduced transition actions result in variability in the number of transition actions. With function word prediction, the number of transition actions for a bag of n words is not necessarily $2n-1$. For example, considering an INSERT, SPLITARC-*to* or SPLITARC-*that* action post each SHIFT action, the maximum number of possible actions is $5n-1$. This variance in the number of actions can impact the linear separability of state items. Following Zhu et al. (2013), we use IDLE actions as a form of padding method, which results in completed state items being further expanded up to $5n-1$ steps. The joint model uses the same perceptron training al-

³For example in Figure 2, *price* is the subject of *be* and if *be* is in present tense and *price* is in plural form, the inflections {*am*, *is*, *was*, *were*} are impossible and *are* is the correct inflection for *be*. We therefore generate transition action as SHIFT-*are*.

Input lemmas: {think₁, price₂, ., increase₄, be₅, have₆, meanwhile₇}

	Transition	σ	ρ	A
0		[1]	{1...7}	\emptyset
1	SH-meanwhile	[7]	{1...6}	
2	IN	[7]	{1...6}	
3	SH-prices	[7 2]	{1,3,4,5,6}	
4	SH-are	[7 2 5]	{1,3,4,6}	
5	SH-thought	[7 2 5 1]	{3,4,6}	
6	SP-to	[7 2 5 1]	{3,4,6}	
7	SH-have	[7 2 5 1 6]	{3,4}	
8	SH-increased	[7 2 5 1 6 4]	{3}	
9	RA	[7 2 5 1 6]	{3}	$A \cup \{6 \rightarrow 4\}$
10	RA	[7 2 5 1]	{3}	$A \cup \{1 \rightarrow 6\}$
11	RA	[7 2 5]	{3}	$A \cup \{5 \rightarrow 1\}$
12	SH-	[7 2 5 3]	{}	
13	RA	[7 2 5]	{}	$A \cup \{5 \rightarrow 3\}$
14	LA	[7 5]	{}	$A \cup \{2 \leftarrow 5\}$
15	LA	[5]	{}	$A \cup \{7 \leftarrow 5\}$

Table 9: Transition action sequence for linearizing the sentence in Figure 2. SH - SHIFT, SP - SPLITARC, RA - RIGHTARC, LA - LEFTARC, IN - INSERT. POS is not shown in SHIFT actions.

gorithm and similar features compared to the baseline model.

4.2 Obtaining Possible Transition Actions Given a Configuration

Given a state $s = ([\sigma | j \ i], \rho, A)$ and an input graph C , the possible transition actions include as a subset the transition actions in Algorithm 1 for shallow graph linearization. In addition, for each lemma being shifted, we enumerate its inflections and create SHIFT transition actions for each inflection. Further, we predict SPLITARC, INSERT and IDLE actions to handle function words. If node i has a child node in C , which is not shifted, we predict SPLITARC and INSERT. If i is sibling to j , we predict INSERT. If both the stack and buffer are empty, we predict IDLE. Pseudocode for GET-POSSIBLEACTIONS for the joint method is shown in Algorithm 5.

5 Experiments

5.1 Dataset

We work on the deep dataset from the Surface Realisation Shared Task (Belz et al., 2011)⁴. Sentences are represented as sets of unordered nodes with labeled semantic edges between them. Semantic representation is obtained by merging Nombank (Meyers et al., 2004), Propbank (Palmer et al., 2005) and syntactic dependencies. Edge labeling follows PropBank annotation scheme such as $\{A0, A1, \dots, An\}$. The nodes are annotated with lemma and where appropriate number, tense and participle features. Function words including

⁴<http://www.nltg.brighton.ac.uk/research/sr-task/>

Algorithm 5: GETPOSSIBLEACTIONS for deep graph linearization, where C is a input graph

Input: A state $s = ([\sigma|j\ i], \rho, A)$ and graph C
Output: A set of possible transition actions T

```

1  $T \leftarrow \emptyset$ 
2 if  $s.\sigma == \emptyset$  then
3   for  $k \in s.\rho$  do
4      $T \leftarrow T \cup (\text{SHIFT}, \text{POS}, k)$ 
5 else
6   if  $\exists k, k \in (\text{DIRECTCHILDREN}(i) \cap s.\rho)$  then
7      $\text{SHIFTSUBTREE}(i, \rho)$ 
8   else
9     if  $A.\text{LEFTCHILD}(i)$  is NIL then
10       $\text{SHIFTSUBTREE}(i, \rho)$ 
11     if  $\{j \rightarrow i\} \in C \wedge A.\text{LEFTCHILD}(j)$  is NIL then
12        $T \leftarrow T \cup (\text{RIGHTARC})$ 
13       if  $i \in \text{DESCENDANT}(j)$  then
14          $\text{PROCESSDESCENDANT}(i, j)$ 
15       if  $i \in \text{SIBLING}(j)$  then
16          $\text{PROCESSSIBLING}(i, j)$ 
17     else if  $\{j \leftarrow i\} \in C$  then
18        $T \leftarrow T \cup (\text{LEFTARC})$ 
19       if  $i \in \text{SIBLING}(j)$  then
20          $\text{PROCESSSIBLING}(i, j)$ 
21     else
22       if  $\text{size}(s.\sigma) == 1$  then
23          $\text{SHIFTPARENTANDSIBLINGS}(i)$ 
24       else
25         if  $i \in \text{DESCENDANT}(j)$  then
26            $\text{PROCESSDESCENDANT}(i, j)$ 
27         if  $i \in \text{SIBLING}(j)$  then
28            $\text{PROCESSSIBLING}(i, j)$ 
29 if  $C.\text{Children}(i) \wedge s.\rho \neq \emptyset$  then
30    $T \leftarrow T \cup (\text{SPLITARC} - to)$ 
31    $T \leftarrow T \cup (\text{SPLITARC} - that)$ 
32 if  $C.\text{Children}(i) \wedge s.\rho \neq \emptyset \vee i \in \text{SIBLING}(j)$  then
33    $T \leftarrow T \cup (\text{INSERT})$ 
34 if  $s.\sigma == \emptyset \wedge s.\rho == \emptyset$  then
35    $T \leftarrow T \cup (\text{IDLE})$ 
36 return  $T$ 

```

that complementizer, *to* infinitive and commas are omitted from the input. There are two punctuation features for information about brackets and quotes. Table 10 shows a sample training instance.

Out of 39k total training instances, 2.8k are non-projective, which we discard. We exclude instances which result in non-projective dependencies mainly because our transition actions predict only projective dependencies. It has been derived from the arc-standard system (Nivre, 2008). There are 1.8k training instances with a mismatch be-

Input (unordered lemma-formed graph):

Sem	ID	PID	Lemma	Attr	Lexeme
SROOT	1	0	be	tense=pres	are
ADV	2	1	meanwhile		meanwhile
P	3	1	.		.
SBJ	4	1	start.02	num=pl	starts
A1	5	4	housing	num=sg	housing
AM-TMP	6	4	september	num=sg	september
VC	9	1	think.01	partic=past	thought
A1	4	9			
C-A1	10	9	have		have
VC	11	10	inch.01	partic=past	inched
A1	4	11			
A5	12	11	upward		upward

Table 10: Deep type training instance from Surface Realisation Shared Task 2011. *Sem* – semantic label, *ID* – unique ID of node within graph, *PID* – the ID of the parent, *Attr* – Attributes such as partic (participle), tense or number, *Lexeme* – lexeme which is resolved using wiktionary and rules in Table 7.

tween edges in the input deep graph and *gold output tree*. The gold output tree is the corresponding shallow tree from the shared task. We approach the task of linearization as extracting a linearized tree from the input semantic graph. So we exclude those instances which do not have edges corresponding to gold tree i.e mismatch between edges of gold tree and input graph. After excluding these instances, we have 34.3k training instances. We also exclude 800 training instances where the function words *to* and *that* have more than one child, and around 100 training instances where function words’ parent and child nodes are not connected by an arc in the deep graph. The above cases are deemed annotation mistakes. We thus train on a final subset of 33.4k training instances. The development set comprises 1034 instances and the test set comprises 2398 instances. Evaluation is done using the BLEU metric (Papineni et al., 2002).

6 Development Results

6.1 Influence of Beam Size

We study the effect of beam size on the accuracies of joint model in Figure 6, by varying the beam size and comparing the accuracies on development dataset over training iterations. Beam sizes of 64 and 128 perform the best. However, beam size 128 does not improve the performance significantly, yet is twice as slow compared to a beam size 64. So we retain a 64 beam for further experiments.

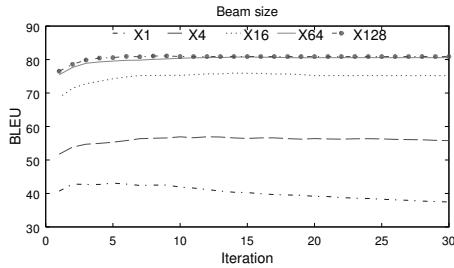


Figure 6: Influence of beam sizes.

	Pipeline	Joint
<i>to</i> infinitive	92.7	94.1
<i>that</i> complementizer	70.6	76.5
count of <i>comma</i>	60.2	63.3

Table 11: Average F-measure for function word prediction for development set.

6.2 Pipeline vs Joint Model

We compare the results of the joint model with the pipeline baseline system. Table 11 shows the development results of function word prediction, and Table 12 shows the overall development results. Our joint model of Transition-Based Deep Input Linearization (TBDIL) achieves an improvement of 5 BLEU points over the pipeline using the same feature source and training algorithm. Thanks to the sharing of word order information, the joint model improves function word prediction compared to the pipeline, which forbids such feature integration because function word prediction is the first step, taken before order becomes available.

7 Final Results

Table 13 shows the final results. The best performing system for the Shared Task was STUMABA-D by Bohnet et al. (2011), which leverages a large-scale n-gram language model. The joint model TBDIL significantly outperforms the pipeline system and achieves an improvement of 1 BLEU point over STUMABA-D, obtaining 80.49 BLEU without making use of external resources.

8 Analysis

Table 14 shows sample outputs from the Pipeline system and the corresponding output from TBDIL. In the first instance, the function word *to* is incorrectly predicted in the arc between nodes *does* and *yield* in the pipeline system. In case of TBDIL, the n-gram feature helps avoid incorrect insertion of *to* which demonstrates the advantage of integrating information across stages. In the second

System	BLEU Score
Pipeline	75.86
TBDIL	80.77

Table 12: Development results.

System	BLEU Score
STUMABA-D	79.43
Pipeline	70.99
TBDIL	80.49

Table 13: Test results.

	output
ref.	if it does n't yield on these matters and eventually begin talking directly to the anc
Pipeline	if it does not to yield on these matters and eventually begin talking directly to the anc
TBDIL	if it does n't yield on these matters and eventually begin talking directly to the anc
ref.	economists who read september 's low level of factory job growth as a sign of a slowdown
Pipeline	september 's low level of factory job growth who as a sign of a slowdown reads economists
TBDIL	economists who read september 's low level of factory job growth as a sign of a slowdown

Table 14: Example outputs.

instance, because of incorrect linearization, there is error propagation to morphological generation in the pipeline system. In particular, *economists* is linearized to the object part of the sentence and the subject is singular. This, in turn, results in the incorrect prediction of morphological form of verb *read* as its singular variant. In TBDIL, in contrast, the joint modelling of linearization and morphology helps ordering the sentence correctly.

9 Conclusion

We showed the usefulness of a joint model for the task of Deep Linearization, by taking (Puduppully et al., 2016) as the baseline and extending it to perform joint graph linearization, function word prediction and morphological generation. To our knowledge, this is the first work to use Transition-Based method for joint NLG from semantic structure. Our system gave the highest scores reported for the NLG 2011 shared task on Deep Input Linearization (Belz et al., 2011).

Acknowledgments

We thank Litton Kurisinkel for helpful discussions and the anonymous reviewers for their detailed and constructive comments. Yue Zhang is supported by the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301.

References

- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, chapter Abstract Meaning Representation for Sembanking, pages 178–186. Association for Computational Linguistics.
- Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*, chapter Evaluation Metrics for Generation.
- Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.
- Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European workshop on natural language generation*, pages 217–226. Association for Computational Linguistics.
- Bernd Bohnet, Leo Wanner, Simon Mille, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 98–106. Association for Computational Linguistics.
- Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. Stumaba: from deep representation to surface. In *Proceedings of the 13th European workshop on natural language generation*, pages 232–235. Association for Computational Linguistics.
- John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Second International Joint Conference on Natural Language Processing: Full Papers*.
- John Carroll, Ann Copestake, and Dan Flickinger. 1999. An efficient chart generator for (semi-) lexicalist grammars.
- Pi-Chuan Chang and Kristina Toutanova. 2007. A discriminative syntactic word order model for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 9–16. Association for Computational Linguistics.
- Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. *Proceedings of the 2014 Conference on EMNLP*, 1:740–750.
- Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.
- Michael Collins. 2002. *Proceedings of the 2002 Conference on EMNLP (EMNLP 2002)*, chapter Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.
- A. De Gispert, M. Tomalin, and W. Byrne. 2014. Word ordering with phrase-based grammars. *14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014*, pages 259–268.
- Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with ccg. In *Proceedings of ACL-08: HLT*, pages 183–191. Association for Computational Linguistics.
- Rose Jenny Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.
- Alice H. Oh and Alexander I. Rudnicky. 2000. *ANLP-NAACL 2000 Workshop: Conversational Systems*, chapter Stochastic Language Generation for Spoken Dialogue Systems.
- Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.
- John Lee and Stephanie Seneff. 2006. Automatic grammar correction for second-language learners. In *INTERSPEECH*, pages 1978–1981.
- Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412. Association for Computational Linguistics.
- Jiangming Liu and Yue Zhang. 2015. An empirical comparison between n-gram and syntactic language models for word ordering. In *Proceedings of the 2015 Conference on EMNLP*, pages 369–378, Lisbon, Portugal, September. Association for Computational Linguistics.
- Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 113–122.

- Igor Aleksandrovič Melčuk. 1988. *Dependency Syntax: theory and practice*. SUNY press.
- Igor Aleksandrovič Melčuk. 2015. *Semantics: From meaning to text*, volume 3. John Benjamins Publishing Company.
- Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating noun argument structure for nombank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA).
- Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the 20th international conference on Computational Linguistics*, page 64. Association for Computational Linguistics.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics, Volume 34, Number 4, December 2008*.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics, Volume 31, Number 1, March 2005*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.
- Ratish Puduppully, Yue Zhang, and Manish Srivastava. 2016. Transition-based syntactic linearization with lookahead features. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 488–493. Association for Computational Linguistics.
- Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren, and Donghong Ji. 2015. A transition-based model for joint segmentation, pos-tagging and normalization. In *Proceedings of the 2015 Conference on EMNLP*, pages 1837–1846, Lisbon, Portugal, September. Association for Computational Linguistics.
- Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(01):57–87.
- Allen Schmaltz, Alexander M. Rush, and Stuart M. Shieber. 2016. Word ordering without syntax. *arXiv preprint arXiv:1604.08633*.
- Lin Feng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1522–1528.
- Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.
- Erik Velldal and Stephan Oepen. 2006. *Proceedings of the 2006 Conference on EMNLP*, chapter Statistical Ranking in Tactical Generation, pages 517–525. Association for Computational Linguistics.
- Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 852–860. Association for Computational Linguistics.
- Michael White and Rajkrishnan Rajkumar. 2009. Perceptron reranking for ccg realization. In *Proceedings of the 2009 Conference on EMNLP*, pages 410–419. Association for Computational Linguistics.
- Michael White. 2004. Reining in ccg chart realization. In *Natural Language Generation*, pages 182–191. Springer Berlin Heidelberg.
- Michael White. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.
- Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on EMNLP*, pages 843–852. Association for Computational Linguistics.
- Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.
- Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.
- Yue Zhang, Kai Song, Lin Feng Song, Jingbo Zhu, and Qun Liu. 2014. Syntactic smt using a discriminative text generation model. In *Proceedings of the 2014 Conference on EMNLP*, pages 177–182, Doha, Qatar, October. Association for Computational Linguistics.
- Yue Zhang. 2013. Partial-tree linearization: generalized word ordering for text synthesis. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2232–2238. AAAI Press.
- Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL (1)*, pages 434–443.

A Obtaining possible transition actions given a configuration for Shallow Graph

During shallow linearization, a state is represented by $s = ([\sigma|j\ i], \rho, A)$ and C is the input graph. Given C , the Decoder outputs actions which extract syntactic tree from the graph. Thus the Decoder outputs RIGHTARC or LEFTARC only if corresponding arc exists in C . The detailed pseudocode is given in Algorithm 1. If i has *direct child nodes* in C , the descendants of i are shifted (line 6-7) (see Algorithm 3). Here, *direct child nodes* (see Algorithm 2) include those child nodes of i for which i is the only parent or if there is more than one parent then every other parent is shifted on to the stack without possibility to reduce the child node. If no *direct child node* is in buffer, then descendants of i are shifted (line 9-10). Now, there are three configurations possible between i and j : 1. i and j are connected by arc in C . This results in RIGHTARC or LEFTARC action; 2. i is descendant of j . In this case the parents of i (such that they are descendants of j) and siblings of i through such parents are shifted. 3. i is sibling of j . In this case, the parents of i and their descendants are shifted such that A remains consistent. Additionally, because the input is a graph structure, more than one of the above configuration can occur simultaneously. We analyse the three configurations in detail below.

Since the *direct child nodes* of i are shifted, $\{j \leftarrow i\}$ results in a LEFTARC action (line 18). Also because the input is a graph, i can be a sibling node of j . In this case, the valid parents and siblings of i are shifted. We iterate through the other elements in stack to identify the valid parents and siblings. These conditions are encapsulated in PROCESSSIBLING (line 20). Conditions for RIGHTARC are similar to that of LEFTARC with the following differences. We ensure that there is no left arc relationship for j in A (line 11). If there is a left arc relationship for j in A , it means that in an arc-standard setting, the RIGHTARC actions for j have already been made. If i is a descendant of j , valid parents and siblings of i are shifted. We iterate through the parents of i and those parents which are in turn descendants of j and not shifted on to the stack are valid parents. We shift the parent and the subtree through each such parent. These conditions are denoted by PROCESSDESCENDANT (line 14).

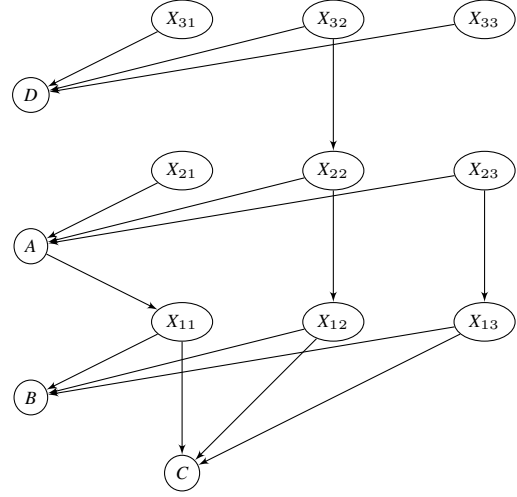


Figure 7: Sample graph to illustrate PROCESSSIBLING

If there is no arc between j and i and there is only one element on the stack, then the parents and siblings of i are shifted (line 22-23). If there is more than one element on the stack, and if i is descendant of j , then we use PROCESSDESCENDANT (line 25-26). If i is sibling to j we use PROCESSSIBLING (line 27-28).

Consider an example to see the working of PROCESSSIBLING in detail. In PROCESSSIBLING, we need to ensure that i is in stack because of sibling relation with j and we need to shift the valid parent nodes of i and their descendants. We call these valid nodes *inflection points*. Consider the following stack entries $[D, A, B, C]$ with C as stack top. Assume that the input graph is as in Figure 7. C is sibling of B through B 's parents X_{11}, X_{12}, X_{13} . Out of these, only X_{11} and X_{12} are valid parents. X_{13} is sibling to A through A 's parent X_{23} . But X_{23} is in turn neither descendant of D nor sibling of D . Thus X_{13} is not a valid inflection point for C . Now, X_{12} is sibling of A through A 's parent X_{22} . X_{22} is in turn sibling of D through X_{32} . Thus there is a path to the stack bottom through a path of siblings/ descendant. In case of X_{11} , X_{11} is descendant of stack element A and is thus valid. X_{11} and X_{12} are called valid inflection points. If inflection point is a common parent to both S_0 and S_1 then inflection point and its descendants are shifted. Instead, if inflection point is ancestor to S_0 , then parents of S_0 (say P_0) which are descendants of inflection point are shifted. Additionally, descendants of P_0 are shifted.

Generating flexible proper name references in text: Data, models and evaluation

Thiago Castro Ferreira and Emiel Krahmer and Sander Wubben

Tilburg center for Cognition and Communication (TiCC)

Tilburg University

The Netherlands

{tcastrof, e. j. krahmer, s. wubben}@tilburguniversity.edu

Abstract

This study introduces a statistical model able to generate variations of a proper name by taking into account the person to be mentioned, the discourse context and variation. The model relies on the REGnames corpus, a dataset with 53,102 proper name references to 1,000 people in different discourse contexts. We evaluate the versions of our model from the perspective of how human writers produce proper names, and also how human readers process them. The corpus¹ and the model² are publicly available.

1 Introduction

In automatic text generation, Referring Expression Generation (REG) is the task responsible for generating references to discourse entities, addressing, for example, the question whether the text should refer to an entity using a definite description (*the West Coast poet and patron saint of drinking writers*), a pronoun (*he*) or a proper name (*Henry Charles Bukowski*). REG is among the tasks which have received most attention in text generation (see Krahmer and van Deemter (2012), for a survey), but the vast majority of the research has concentrated on the generation of descriptions, while proper name generation has received virtually no attention, albeit with notable exceptions (Siddharthan et al., 2011; van Deemter, 2016) to which we return below.

Still, proper names occur frequently in texts. For instance, Ferreira et al. (2016a) showed that human writers use proper names in 91% of the cases to initially refer to persons. Indeed, some

earlier research on text generation has stated that discourse-new references should be generated by using the strategy to “simply give the name of the object (if it has a name)” (Reiter and Dale, 2000). However, the *Bukowski* example already indicates that this is not as straightforward as Reiter and Dale suggest - the poet’s full name is *Henry Charles Bukowski* and his birth name is *Heinrich Karl Bukowski*, but he is more commonly known as simply *Charles Bukowski*; see also van Deemter (2016), for a discussion of this and other complicating factors in proper name generation. In addition, Reiter and Dale (2000) do not address how repeated references using a name in a text should be generated. For instance, should our discourse-old example-writer be referred to as *Charles*, *Bukowski* or some combination of these and other attributes (e.g., using a modifier like *the poet Bukowski*)?

Imagine, for the sake of argument, that we would generate proper name references in a text by initially generating the full name, after which repeated references only consist of the last name (a.k.a. the family or surname). Intuitively, it is not difficult to come up with counterexamples to this “rule”. Above we already discussed the difficulties of deciding what the most appropriate full name reference is for *Henry Charles Bukowski*, which (like *Keith Rupert Murdoch* and *Walter Bruce Willis*) seems to be the combination of middle and last names (as opposed to *Oprah Gail Winfrey* and *Serena Jameka Williams*, for who it is more common the combination of first and last names). Moreover, using the last name for repeated references may work well for the likes of *Winston Churchill* and *Angela Merkel*, but seems less suitable for *Napoleon Bonaparte* or *Madonna Ciccone*, to mention just two. Moreover, our example rule cannot account for the occurrence of modifiers. And, finally, it seems highly unlikely

¹<http://ilk.uvt.nl/~tcastrof/regnames/>

²<http://github.com/ThiagoCF05/ProperName>

that human writers would adhere to such a strict rule. Rather, one might expect writers to vary in their choices of which name to use, depending on stylistic and discourse factors, much like the choice of referential form varies as a function of such factors (Ferreira et al., 2016a; Ferreira et al., 2016b).

In general, we know very little about how proper names should be generated in text – as far as we know, there have been hardly any systematic corpus studies and only very little concrete proposals on how to automatically generate proper name references. In this paper, we therefore present a large scale corpus analysis, and, based on this, two versions of a new probabilistic model of proper name generation: one that always chooses the most likely proper name form and one that relies on a ‘roulettwheel’ selection model and hence will generate more varied references. These models rely both on the nature of the entity referred to (what is the likelihood that a given person will be referred to using, say, the first or last name?) and on the discourse context for generating proper name references in text. In an intrinsic evaluation experiment, we compare the performance of the two versions of this model with our implementations of the two proposals that have been made before (Siddharthan et al., 2011; van Deemter, 2016). We also describe a human evaluation experiment where we compare original texts with alternative versions that include proper names generated by our model.

2 Related work

Even though proper name references occur frequently in written text, their generation remains seriously understudied. A recent survey of REG models (Krahmer and van Deemter, 2012) has essentially nothing to say about the topic, and general surveys of automatic text generation such as Reiter and Dale (2000) only briefly mention a very basic rule (use a proper name, if available, for first references), without further specifying or evaluating it.

Recently, van Deemter (2016) has highlighted the importance of proper name generation. After discussing why a simple rule like the one proposed by Reiter and Dale cannot account for the complexities of proper name references in text, he argues that names could just be treated like other attributes in the generation of descriptions. Put dif-

ferently, the name of an object can be modelled just like its color or size (typical attributes used in REG examples) – just as a description like *the tall man* rules out men that are not tall, so does a proper name like *Charles* rule out other people not named Charles. A standard REG algorithm, such as, for example, the Incremental Algorithm (Dale and Reiter, 1995) can then be used to compute when a name should be used and in which form. Van Deemter’s work is of a theoretical nature; he has not implemented or tested this idea, so we cannot tell how well it can account for proper name references in text. In addition, in this form, his proposal cannot account for possible variations in proper name form throughout a text.

The most detailed study of proper name generation, as far as we know, is the seminal study by Siddharthan et al. (2011), which (re-)generates references to people in news summaries. For their algorithm(s), the authors present two manually constructed rules, based on earlier theories of reference, one for discourse-new references (including the full name) and one for discourse-old references (which in full says: “Use surname only, remove all pre- and post-modifiers.”). They discuss, based on corpus analyses, how notions like discourse-new and discourse-old can be learned without manual annotation, and how they co-determine whether additional attributes such as role and affiliation should be included. Finally, they show that their model leads to improved (more coherent) summaries. While the approach offers a very interesting solution for the generation of discourse-new proper name references with modifiers for major characters in a news story (*Former East German leader Erich Honecker*), the proper name generation rule itself is very similar to the example rule discussed in the introduction (use the full name for discourse-new references and only the surname for discourse-old references). It is not specified how the full name should be realised (remember the *Henry Charles Bukowski*-example), and neither can the approach deal with exceptions to the surname-only rule (remember the *Madonna Ciccone*-example) or with intratext variation.

3 REGnames

For our explorations, we relied on the REGnames corpus (Ferreira et al., 2016c). REGnames is a corpus of 53,102 proper names referring to 1,000

people in 15,241 texts. The corpus consists of webpages extracted from the Wikilinks corpus (Singh et al., 2012), which was initially collected for the study of cross-document coreference and consists of more than 40 million references to almost 3 million entities in around 11 million webpages. All the references annotated in Wikilinks were grouped according to the Wikipedia page of the entity. This procedure enables easy identification of the mentioned entity and facilitates the extraction of more information about it.

To build the REGnames corpus, Ferreira et al. (2016c) selected the 1,000 most frequently mentioned people in the Wikilinks corpus. Then for each person, they selected random webpages from Wikilinks which mention the person at least once. On all selected webpages, part-of-speech tagging, lemmatization, named entity recognition, dependency parsing, syntactic parsing, sentiment analysis and coreference resolution was performed by using the Stanford CoreNLP software (Manning et al., 2014).

All extracted proper names were automatically annotated with their syntactic position (subject, object or genitive noun phrase in a sentence) and referential statuses in the text (discourse-new or discourse-old) and in the sentence (sentence-new or sentence-old). The extracted proper names were also annotated according to their form, i.e. which kind(s) of name (first, middle and/or last names), and modifier(s) (title and/or appositive) were part of the proper name. To check for the presence of first, middle and last names, a Proper Name Knowledge Base was extracted from DBpedia (Bizer et al., 2009) with all the names of the people in the corpus. Then, to check for the presence of a title or an appositive, named entity recognition information and the dependency tree were used respectively.

In the corpus analysis, Ferreira et al. (2016c) noticed that proper name references generally decrease in lengths across the text. They also concluded that a discourse-old or sentence-new proper name reference in the object position of a sentence tends to be shorter than a discourse-new or sentence-old proper name reference in the subject position of a sentence. In general, the corpus is a valuable resource which can be used to train a statistical model for proper name generation, as we show in the next section.

4 A model for proper name generation

Similarly to the generation of definite descriptions, our model produces a proper name reference in two sequential steps: content selection and linguistic realization.

4.1 Content Selection

The content selection discussed here is analogous to the selection of semantic attributes (type, color, size, etc) when generating a description of an entity (Dale and Haddock, 1991; Dale and Reiter, 1995). However, instead of attributes, the content selection step in our model aims to choose the *form* of a proper name reference (which kind(s) of name and modifier(s) are part of the proper name reference).

Features By analysing the REGnames corpus, Ferreira et al. (2016c) observed that proper names vary in their forms throughout a text. Moreover, as discussed in the Introduction (Section 1), a proper name form can also be influenced by the person to be mentioned. Thus, we conditioned the choice of a specific proper name form by a set of discourse features that describe the reference as well as to the person to be mentioned.

Table 1 depicts the discourse features used to describe the proper name references. We choose them based on the analysis of the REGnames corpus (Section 3).

Forms Our model selects a proper name form over all forms annotated on the REGnames corpus, i.e. a total of 28 possible ones. Table 2 depicts the most frequent ones. The complete list can be found at the webpage that describes the REGnames corpus³.

Notation Given a person p to be referred to by his/her proper name and the set of discourse features D that describe the reference, we aim to predict the form $f \in F$ of a proper name as Equation 1 shows.

$$P(f | D, p) = \frac{P(f | p) \prod_{d \in D} P(d | f, p)}{\sum_{f' \in F} P(f' | p) \prod_{d \in D} P(d | f', p)} \quad (1)$$

To account for unseen data, the conditional probabilities are computed using the additive

³<http://ilk.uvt.nl/~tcastrof/regnames/>

Feature	Description
Syntactic Position	Subject, object or a genitive noun phrase in the sentence.
Referential Status	First mention of the referent (new) or not (old) at the level of text and sentence.

Table 1: Discourse features that describe the references.

smoothing technique with $\alpha = 1$. Equations 2 and 3 summarize the procedure.

$$P(f | p) = \frac{\text{count}(f \cap p) + \alpha}{\text{count}(p) + \alpha|F|} \quad (2)$$

$$P(d | f, p) = \frac{\text{count}(d \cap f \cap p) + \alpha}{\text{count}(f \cap p) + \alpha|D|} \quad (3)$$

Variation Besides the fact that proper name references may vary in their forms throughout a text and according to the person to be referred to, they may also vary in similar situations of a text. In an extrinsic evaluation comparing human- and machine-generated summaries, for instance, Sidharthan et al. (2011) reported that the lack of variation in the form of discourse-old proper names references was one of the disadvantages of their summarization system in the cases where human summaries were chosen. Our model fills this gap by performing Equation 1 over all the proper name forms given a set of similar references. That is proper name references to the same person and described by the same set of discourse feature values. This procedure results in a frequency distribution over all relevant proper name forms. Then, similar to the roulettewheel selection of Ferreira et al. (2016b) for the choice of referential forms, we can randomly apply the frequencies into a group of similar references in such a way that their forms will be representative of the distribution predicted by the model. For instance, given a group of 5 references and a frequency distribution of 0.8 for the *first+last* form and 0.2 for the *last* form, 4 references would assume the first form, whereas 1 reference would assume the other one.

4.2 Linguistic Realization

Once we select the form of a proper name reference to a person in a particular discourse context, we linguistically realize this reference by choosing the most likely words - including titles and proper nouns - to be part of it. The process is analogous to the linguistic realization of a set of attribute-values into a description (Bohnet, 2008; Zarriess and Kuhn, 2013). Equation 4 summarizes it.

Form	Frequency
First+Last	46.2%
Last	34.9%
First	8.5%
Middle+Last	2.8%
First+Middle+Last	2.3%
Middle	1.5%
Others	3.5%

Table 2: Most popular proper name forms in REG-names corpus and their frequencies.

$$P(n_1 \dots n_t | f, p) = \prod_t P(n_t | n_{t-1}, \{e_i\}_{i=1}^{|f|}, p) \quad (4)$$

The vocabulary used in the linguistic realization step consists of all the titles found in REGnames, all the possible names of the given person present in the corpus’ proper name knowledge base, and an *end* token, present at the end of all proper name references in the training set. The process finishes when this token is predicted ($n_t = END$). The choice of a word n_t is conditioned to the previous generated word in the proper name reference (n_{t-1}), the elements present in the given form ($\{e_i\}_{i=1}^{|f|}$: constrained to first, middle and last name; plus title and appositive) and the person to be referred to (p). If $P(n_t | n_{t-1}, \{e_i\}_{i=1}^{|f|}, p) = 0$, we drop the less frequent element from the given proper name form. If all the elements were dropped and the probability would still be 0, we conditioned the choice only to the person ($P(n_t | p)$). Regarding the cases in which the original proper name form indicates the presence of an appositive, we add a description - obtained from Wikidata (Vrandečić and Krötzsch, 2014) - at the end of the generated proper name reference.

5 Baselines

In order to evaluate the performance of our model, we developed three baseline models. All the models have their outputs constrained to three choices: given name, surname and full name of a person.

Given name and surname are determined by the values of the following attributes in the person’s DBpedia page: *foaf:givenName* and *foaf:surname*. Full name was defined as the combination of both values. If these attributes are missing, we use the birth name of the person, also extracted from DBpedia (*dbp:birthName*). In this situation, the full name of a person will be the proper birth name, whereas given and surnames will be the first and last tokens from the birth name, respectively.

The first baseline, called *Random*, is a baseline that randomly chooses one of the three options to generate a proper name.

The second baseline is an adaptation of the model proposed by van Deemter (2016) and will be called *Deemter*. Among the full name, given name and surname of a person, our adaptation chooses the shortest name that distinguishes the mentioned person from all other entities in the current and previous 3 sentences in the text. It is important to stress that this model is our adaptation, since the proposal of van Deemter (2016) only applies for initial references, not for repeated ones in a text.

Finally, the third system we compare against is based on Siddharthan et al. (2011) and will be called *Siddharthan*. This baseline chooses the full name of a person for discourse-new references; and his/her surname otherwise.

6 Automatic Evaluation

We intrinsically evaluate the models by training and testing them on a subset of the REGnames corpus. This evaluation aims to investigate how close our model can produce proper name references to the ones generated by human writers.

6.1 Data

We considered a subset of the REGnames corpus as our evaluation data. From the 1,000 people in the corpus, we first filtered the ones whose birth names were not mentioned, or for whom the values of the DBpedia’s attributes *foaf:name*, *foaf:givenName* and *foaf:surname* were missing. This measure was taken in order to have a consistent vocabulary to linguistically realize the proper name references, as well as to make sure that our baselines would always have a consistent output. Then, from the remaining people, we only selected the ones with at least 50 proper name references in the REGnames corpus such that we could train and

test our model properly. In total, we used 43,655 proper names references to 432 people as our evaluation data.

In order to investigate the influence of the text domain in the generation of proper names, we classified the webpages from where our evaluation data were extracted according to 3 domains: Blog, News and Wiki. All the webpages whose the url contained the substrings *blog*, *tumblr* or *wordpress* were classified as part of the blog domain. If the substrings were *new* or *article*, the webpage was classified as a news. Finally, we classified as Wiki all the webpages whose the url contained the substring *wiki*. All the other webpages were grouped into a *Other domains* category.

6.2 Method

10-fold-cross-validation was performed to evaluate the models. We made sure that the number of references per person was uniform among the folds. To measure the models performance in the choice of the proper name form, accuracy was used. To check the similarity among the realized proper name reference and the gold standard one, we used the string edit distance.

6.3 Models

We evaluated the three proposed baselines (*Random*, *Deemter* and *Siddharthan*) and two versions of our model: *PN-Variation* and *PN+Variation*.

PN-Variation does not take the variation into account in the content selection. In other words, this model always chooses the most likely proper name form for the references in the test set which refer to the same person and are described by the same combination of discourse feature values. On the other hand, *PN+Variation* takes variation into account by applying the distribution of proper name forms obtained from the training set to the similar references in the test set, as explained in Section 4.1.

6.4 Results

Table 3 summarizes the accuracy-scores of the models in the prediction of the proper name forms. Both versions of our model outperform the baselines for all the domains. *PN-Variation* is the model with the highest accuracy.

Figure 1 depicts the string edit distance among the gold standard proper names and the ones generated by the proposed models. A Repeated Measures ANOVA determined that the string edit dis-

Model	Blog	News	Wiki	Other domains	Overall
Random	0.25	0.22	0.22	0.25	0.25
Deemter	0.33	0.30	0.28	0.33	0.33
Siddharthan	0.52	0.48	0.42	0.45	0.48
PN-Variation	0.66	0.63	0.66	0.70	0.68
PN+Variation	0.58	0.55	0.59	0.63	0.60

Table 3: Proper name form accuracies of our two models (PN-Variation and PN+Variation) as a function of text genre and compared to three baseline models (Random, Deemter, Siddharthan).

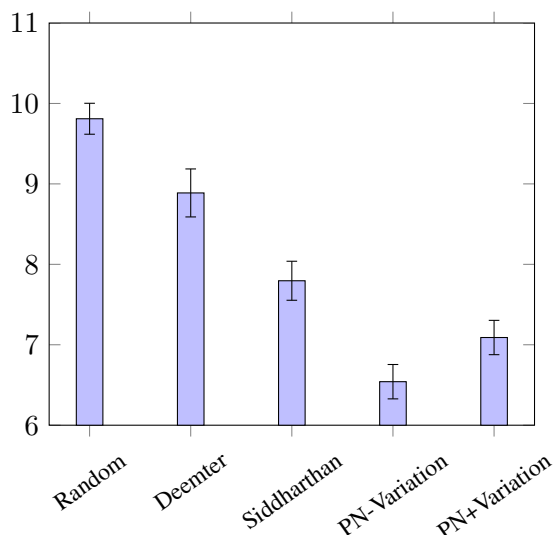


Figure 1: String edit distance in the overall corpus. Error bars represent 95% confidence intervals.

tances of the models were significantly different ($F(4, 36) = 1630, p < .001$). We performed a post hoc analysis with paired t-test using Bonferroni adjusted alpha levels of 0.005 per test (0.05/10). Both versions of our model significantly outperform the baselines with all pairwise comparisons significant at $p < .001$. Regarding the comparison of our models, *PN-Variation* is significantly better than *PN+Variation* ($t(9) = -38.14, p < .001$).

Figure 2 shows the evaluation of our models by domain. A Repeated Measures ANOVA shows that the string edit distances of the models are significantly different in all domains (Blog: $F(4, 36) = 718.8, p < .001$; News: $F(4, 36) = 308.2, p < .001$; Wiki: $F(4, 36) = 118.5, p < .001$; Other domains: $F(4, 36) = 2213, p < .001$).

We also performed a post hoc analysis for the results by domain in the same style we did for the general results. In the blog and news do-

main, both versions of our model significantly outperform all the baselines with all pairwise comparisons significant at $p < .005$. Among our models, *PN-Variation* is significantly better than *PN+Variation* (Blog: $t(9) = -26.33, p < .001$; News: $t(9) = -7.45, p < .001$).

In the wiki domain and in texts which are not part of the blog, news and wiki domain, both versions of our model also significantly outperform all the baselines with all pairwise comparisons significant at $p < .001$. The difference in the results of *PN-Variation* and *PN+Variation* is also significant (Wiki: $t(9) = -4.91, p < .001$; Other domains: $t(9) = -27.14, p < .001$).

7 Human Evaluation

We also performed a human evaluation aiming to compare original texts with alternative versions whose proper name references were generated by our model. This evaluation aims to investigate the quality of the proper name references from the perspective of the human reader.

7.1 Materials

We used 9 abstracts from English Wikipedia pages whose topic is one of the people studied in the REGnames corpus. They were extracted from DBpedia and have at least 10 proper name references to the topic.

Although our model did not yield its best results for this domain, it was chosen based on the relatively short length of the texts and the large amount of proper name references they have. Moreover, the proper name references in Wikipedia abstracts are similar to the ones generated by our *Siddharthan* baseline, i.e. a full name to discourse-new people, and surname to discourse-old people.

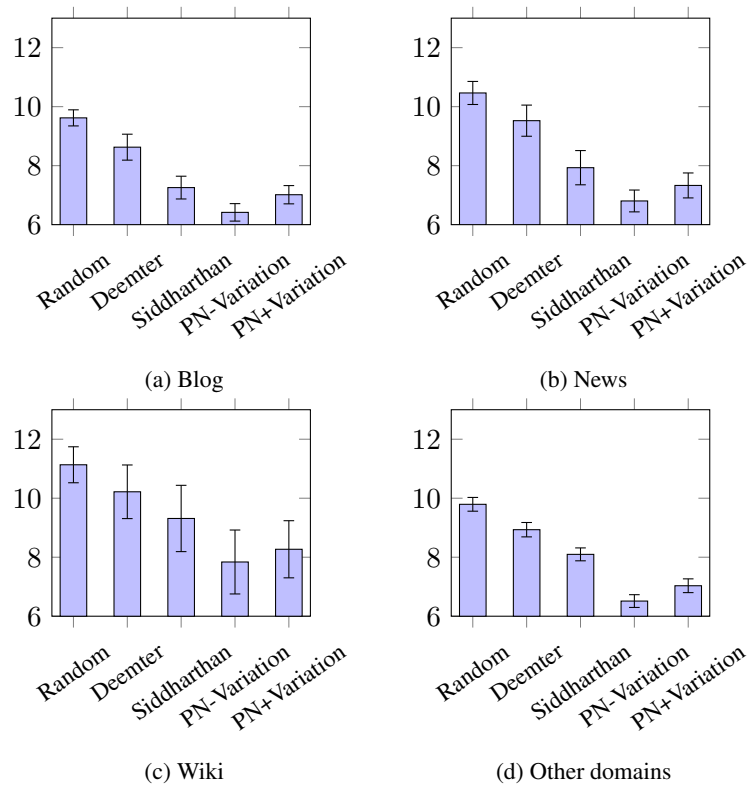


Figure 2: String edit distances of the models in the (2a) blog, (2b) news, (2c) wiki and (2d) in other domains which are not the previous ones. Error bars represent 95% confidence intervals.

7.2 Method

For each abstract, we designed 3 trials. In the first, we presented participants with the original text next to the version with the proper name references generated by the *PN-Variation* model (Original vs. No Variation). In the second, we presented the original text next to the version with the proper name references generated by the *PN+Variation* (Original vs. Variation). Finally, the third trial consists of the text versions with the proper name references produced by both versions of our model (No Variation vs. Variation). The trials of a text were distributed in different lists such that we obtained 3 lists with 9 texts - 3 trials of each type in a list. In all the texts, the proper name references were highlighted in yellow. For each trial, we asked participants to choose which text they preferred, taking into account the highlighted references. The experiment is publicly available⁴.

We recruited 60 participants through Crowdfunder – 20 per list. Of the participants, 44 were female and their average age was 36 years. All participants reported to be proficient in the English language (58 were native speakers).

⁴<http://ilk.uvt.nl/~tcastrof/eacl2017>

7.3 Results

The texts of the “Original” version were the favourite of 69% of the participants in comparison with texts of the “No Variation” version (Chi-square $\chi^2(2, 180) = 25.69, p < .001$), and 75% participants with the “Variation” version (Chi-square $\chi^2(2, 180) = 45; p < .001$). Regarding the “No Variation vs. Variation” trials, texts of the “No Variation” version were the favourite of the participants in 59% of the cases (Chi-square $\chi^2(2, 180) = 6.42; p < .05$).

8 General Discussion

Proper name generation is a seriously understudied phenomenon in automatic text generation. There are many different ways in which a person can be referred to in a text using their name (*Barack Hussein Obama II, Barack Obama, Obama, President Obama, etc.*) and arguably a text that uses different naming formats in different conditions is more human-like than one that relies on a fixed strategy (e.g., always use the full name).

This paper introduced a new statistical model for the generation of proper names in text, taking into account three different factors: (1) who

the person is, (2) in which discourse context the proper name reference should be generated and (3) the different forms that a proper name can assume in similar situations (variation). The model was developed based on the REGnames corpus (Ferreira et al., 2016c), which contains a large number of proper name references in various discourse situations. We also implemented two other systems for the sake of comparison: one based on the Siddharthan et al. (2011) model and one based on the ideas for proper name reference proposed by van Deemter (2016).

We developed two versions of our model: one that deterministically generated the best proper name form in a given setting (*PN-variation*), and one that relied on a probabilistic distribution over different forms, allowing for more variation in the output (*PN+Variation*). Both models were systematically compared to a random baseline and the two alternative models due to Siddharthan et al. (2011) and van Deemter (2016).

Automatic Evaluation We first conducted an automatic evaluation investigating to what extent the evaluated models produced proper name references similar to the ones generated by human writers, using a held-out subset of the REGnames corpus. In general, we found that both versions of our model were able to outperform a random baseline and the two reference systems, where the version without variation (*PN-Variation*) yielded the best results. Across text domains, there was variation in the performance of both versions of our model. The worst results were registered in the Wiki domain, suggesting that text domain is a factor that may be taken into account in the task of generating proper names.

Human Evaluation In the automatic evaluation experiment, the differences between the system with and without variation were small, so in a second study we asked whether human readers preferred the output from one of these systems over the other. For this purpose, we conducted an experiment consisting of pairwise comparisons based on texts taken from the Wikipedia domain, where we compared the output produced by the *PN-variation* and the *PN+variation* system with the original text and also among them. Interestingly, we found that people had a general preference for the no-variation model over the one that non-deterministically generated varied texts. This

suggests that readers prefer consistency in proper name references to the same topic in similar situations, which is different from the choice of referential *form* (Ferreira et al., 2016b).

Additionally, we found that participants preferred the original over the regenerated texts. We suspect that this preference was due to the initial discourse-new proper name reference, which in the Wikipedia texts has a special status. Usually, the initial reference to the topic is not the most common proper name reference in other domains, but a specific Wikipedia format which our system does not produce. For example, the original text about Magic Johnson starts with *Earvin “Magic” Johnson Jr.* in the discourse-new proper name reference, while our system simply produced *Magic Johnson*.

Semantic web Earlier work on REG models has concentrated on the generation of descriptions, typically assuming the existence of a knowledge base of entities (Dale and Haddock, 1991; Dale and Reiter, 1995) or introducing one to small domains (Gatt and Belz, 2010). Our REG models for proper names, however, strongly rely on the semantic web as an information resource of the entities to be referred to. Databases like DBpedia (Bizer et al., 2009) and Wikidata (Vrandečić and Krötzsch, 2014) provide information about thousands of entities and can be used in different domains.

Baselines We developed two powerful baselines based on proposals that have been made before. *Deemter* (van Deemter, 2016) relies on the criteria of the first developed REG models (Dale and Haddock, 1991; Dale and Reiter, 1995): given a target, produce a reference that distinguishes it from the distractors in the context. Our model as presented does not make this assumption (it does not always produce a proper name reference that distinguishes the target from the distractors). However, this could be incorporated into our model as well. For instance, given a list of the most likely proper name references produced by our model in a situation, we can choose the one with the highest likelihood that distinguishes the target from all other entities in the current and previous 3 sentences in the text (as in the *Deemter* model).

Regarding performance, *Siddharthan* is the baseline that performed best. The original version, proposed in Siddharthan et al. (2011), is

even able to decide whether to include a modifier in a discourse-new reference based on the global salience of the entity mentioned. However, the model is arguably more limited in the production of a proper name itself. By always generating a surname in discourse-old references for instance, the Siddharthan model is not able to generate at least 10% of the references in the REGnames corpus (8.5% consist of *first name* references, and 1.5% of *middle name* ones).

Conclusion In sum, we conclude that our model is able to generate proper name references similar to the ones produced by human writers. In future research, it would be interesting to further investigate the role of text genre in proper name references as well as the influence of variation on proper name forms.

Acknowledgments

This work has been supported by the National Council of Scientific and Technological Development from Brazil (CNPq). We would also like to thank the members of the Language Production group at TiCC, specially Ákos Kádár, for insightful comments on the manuscript.

References

- Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165. The Web of Data.
- Bernd Bohnet. 2008. The fingerprint of human referring expressions and their surface realization with graph transducers. In *Proceedings of the Fifth International Natural Language Generation Conference, INLG '08*, pages 207–210, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Robert Dale and Nicholas Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of the fifth conference on European chapter of the Association for Computational Linguistics, EACL '91*, pages 161–166, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.
- Thiago Castro Ferreira, Emiel Kraemer, and Sander Wubben. 2016a. Individual variation in the choice of referential form. In *Proceedings of NAACL-HLT*, pages 423–427, San Diego, California. Association for Computational Linguistics.
- Thiago Castro Ferreira, Emiel Kraemer, and Sander Wubben. 2016b. Towards more variation in text generation: Developing and evaluating variation models for choice of referential form. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 568–577, Berlin, Germany. Association for Computational Linguistics.
- Thiago Castro Ferreira, Sander Wubben, and Emiel Kraemer. 2016c. Towards proper name generation: a corpus analysis. In *Proceedings of the 9th International Natural Language Generation conference (INLG)*, Edinburgh, Scotland.
- Albert Gatt and Anja Belz. 2010. Introducing shared tasks to nlg: The tuna shared task evaluation challenges. In *Empirical methods in natural language generation*, pages 264–293. Springer.
- Emiel Kraemer and Kees van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.
- Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.
- Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015.
- Kees van Deemter. 2016. Designing algorithms for referring with proper names. In *Proceedings of the 9th International Natural Language Generation conference*, pages 31–35, Edinburgh, UK, September 5-8. Association for Computational Linguistics.
- Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September.
- Sina Zarriess and Jonas Kuhn. 2013. Combining Referring Expression Generation and Surface Realization: A Corpus-Based Investigation of Architectures. In *Proceedings of the 51st Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 1547–1557, Sofia, Bulgaria, August. Association for Computational Linguistics.

Dependency Parsing as Head Selection

Xingxing Zhang, Jianpeng Cheng and Mirella Lapata

Institute for Language, Cognition and Computation

School of Informatics, University of Edinburgh

10 Crichton Street, Edinburgh EH8 9AB

{x.zhang, jianpeng.cheng}@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Conventional graph-based dependency parsers guarantee a tree structure both during training and inference. Instead, we formalize dependency parsing as the problem of independently selecting the head of each word in a sentence. Our model which we call DENSE (as shorthand for **D**ependency **N**eural **S**election) produces a distribution over possible heads for each word using features obtained from a bidirectional recurrent neural network. Without enforcing structural constraints during training, DENSE generates (at inference time) trees for the overwhelming majority of sentences, while non-tree outputs can be adjusted with a maximum spanning tree algorithm. We evaluate DENSE on four languages (English, Chinese, Czech, and German) with varying degrees of non-projectivity. Despite the simplicity of the approach, our parsers are on par with the state of the art.¹

1 Introduction

Dependency parsing plays an important role in many natural language applications, such as relation extraction (Fundel et al., 2007), machine translation (Carreras and Collins, 2009), language modeling (Chelba et al., 1997; Zhang et al., 2016) and ontology construction (Snow et al., 2005). Dependency parsers represent syntactic information as a set of head-dependent relational arcs, typically constrained to form a tree. Practically all models proposed for dependency parsing in recent years can be described as graph-based (McDon-

ald et al., 2005a) or transition-based (Yamada and Matsumoto, 2003; Nivre et al., 2006b).

Graph-based dependency parsers are typically arc-factored, where the score of a tree is defined as the sum of the scores of all its arcs. An arc is scored with a set of local features and a linear model, the parameters of which can be effectively learned with online algorithms (Crammer and Singer, 2001; Crammer and Singer, 2003; Freund and Schapire, 1999; Collins, 2002). In order to efficiently find the best scoring tree during training *and* decoding, various maximization algorithms have been developed (Eisner, 1996; Eisner, 2000; McDonald et al., 2005b). In general, graph-based methods are optimized globally, using features of single arcs in order to make the learning and inference tractable. Transition-based algorithms factorize a tree into a set of parsing actions. At each transition state, the parser scores a candidate action conditioned on the state of the transition system and the parsing history, and greedily selects the highest-scoring action to execute. This score is typically obtained with a classifier based on non-local features defined over a rich history of parsing decisions (Yamada and Matsumoto, 2003; Zhang and Nivre, 2011).

Regardless of the algorithm used, most well-known dependency parsers, such as the MST-Parser (McDonald et al., 2005b) and the MaltParser (Nivre et al., 2006a), rely on extensive feature engineering. Feature templates are typically manually designed and aim at capturing head-dependent relationships which are notoriously sparse and difficult to estimate. More recently, a few approaches (Chen and Manning, 2014; Lei et al., 2014a; Kiperwasser and Goldberg, 2016) apply neural networks for learning dense feature representations. The learned features are subsequently used in a conventional graph- or transition-based parser, or better de-

¹Our code is available at http://github.com/XingxingZhang/dense_parser.

signed variants (Dyer et al., 2015).

In this work, we propose a simple neural network-based model which learns to select the head for each word in a sentence without enforcing tree structured output. Our model which we call DENSE (as shorthand for **Dependency Neural Selection**) employs bidirectional recurrent neural networks to learn feature representations for words in a sentence. These features are subsequently used to predict the head of each word. Although there is nothing inherent in the model to enforce tree-structured output, when tested on an English dataset, it is able to generate trees for 95% of the sentences, 87% of which are projective. The remaining non-tree (or non-projective) outputs are post-processed with the Chu-Liu-Edmond (or Eisner) algorithm. DENSE uses the head selection procedure to estimate arc weights during training. During testing, it essentially reduces to a standard graph-based parser when it fails to produce tree (or projective) output.

We evaluate our model on benchmark dependency parsing corpora, representing four languages (English, Chinese, Czech, and German) with varying degrees of non-projectivity. Despite the simplicity of our approach, experiments show that the resulting parsers are on par with the state of the art.

2 Related Work

Graph-based Parsing Graph-based dependency parsers employ a model for scoring possible dependency graphs for a given sentence. The graphs are typically factored into their component arcs and the score of a tree is defined as the sum of its arcs. This factorization enables tractable search for the highest scoring graph structure which is commonly formulated as the search for the maximum spanning tree (MST). The Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005b) is often used to extract the MST in the case of non-projective trees, and the Eisner algorithm (Eisner, 1996; Eisner, 2000) in the case of projective trees. During training, weight parameters of the scoring function can be learned with margin-based algorithms (Crammer and Singer, 2001; Crammer and Singer, 2003) or the structured perceptron (Freund and Schapire, 1999; Collins, 2002). Beyond basic first-order models, the literature offers a few examples of

higher-order models involving sibling and grand parent relations (Carreras, 2007; Koo et al., 2010; Zhang and McDonald, 2012). Although more expressive, such models render both training and inference more challenging.

Transition-based Parsing As the term implies, transition-based parsers conceptualize the process of transforming a sentence into a dependency tree as a sequence of transitions. A transition system typically includes a stack for storing partially processed tokens, a buffer containing the remaining input, and a set of arcs containing all dependencies between tokens that have been added so far (Nivre, 2003; Nivre et al., 2006b). A dependency tree is constructed by manipulating the stack and buffer, and appending arcs with predetermined operations. Most popular parsers employ an *arc-standard* (Yamada and Matsumoto, 2003; Nivre, 2004) or *arc-eager* transition system (Nivre, 2008). Extensions of the latter include the use of non-local training methods to avoid greedy error propagation (Zhang and Clark, 2008; Huang and Sagae, 2010; Zhang and Nivre, 2011; Goldberg and Nivre, 2012).

Neural Network-based Features Neural network representations have a long history in syntactic parsing (Mayberry and Miikkulainen, 1999; Henderson, 2004; Titov and Henderson, 2007). Recent work uses neural networks in lieu of the linear classifiers typically employed in conventional transition- or graph-based dependency parsers. For example, Chen and Manning (2014) use a feed forward neural network to learn features for a transition-based parser, whereas Lei et al. (2014a) do the same for a graph-based parser. Lei et al. (2014b) apply tensor decomposition to obtain word embeddings in their syntactic roles, which they subsequently use in a graph-based parser. Dyer et al. (2015) redesign components of a transition-based system where the buffer, stack, and action sequences are modeled separately with stack long short-term memory networks. The hidden states of these LSTMs are concatenated and used as features to a final transition classifier. Kiperwasser and Goldberg (2016) use bidirectional LSTMs to extract features for a transition- and graph-based parser, whereas Cross and Huang (2016) build a greedy arc-standard parser using similar features.

In our work, we formalize dependency parsing

as the task of finding for each word in a sentence its most probable head. Both head selection and the features it is based on are learned using neural networks. The idea of modeling child-parent relations independently dates back to Hall (2007) who use an edge-factored model to generate k -best parse trees which are subsequently reranked using a model based on rich global features. Later Smith (2010) show that a head selection variant of their loopy belief propagation parser performs worse than a model which incorporates tree structure constraints. Our parser is conceptually simpler: we rely on head selection to do most of the work and decode the best tree *directly* without using a reranker. In common with recent neural network-based dependency parsers, we aim to alleviate the need for hand-crafting feature combinations. Beyond feature learning, we further show that it is possible to simplify the training of a graph-based dependency parser in the context of bidirectional recurrent neural networks.

3 Dependency Parsing as Head Selection

In this section we present our parsing model, DENSE, which tries to predict the head of each word in a sentence. Specifically, the model takes as input a sentence of length N and outputs N ⟨head, dependent⟩ arcs. We describe the model focusing on unlabeled dependencies and then discuss how it can be straightforwardly extended to the labeled setting. We begin by explaining how words are represented in our model and then give details on how DENSE makes predictions based on these learned representations. Since there is no guarantee that the outputs of DENSE are trees (although they mostly are), we also discuss how to extend DENSE in order to enforce projective and non-projective tree outputs. Throughout this paper, lowercase boldface letters denote vectors (e.g., \mathbf{v} or \mathbf{v}_i), uppercase boldface letters denote matrices (e.g., \mathbf{M} or \mathbf{M}_b), and lowercase letters denote scalars (e.g., w or w_i).

3.1 Word Representation

Let $S = (w_0, w_1, \dots, w_N)$ denote a sentence of length N ; following common practice in the dependency parsing literature (Kübler et al., 2009), we add an artificial ROOT token represented by w_0 . Analogously, let $A = (\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_N)$ denote the representation of sentence S , with \mathbf{a}_i representing word w_i ($0 \leq i \leq N$). Besides encoding infor-

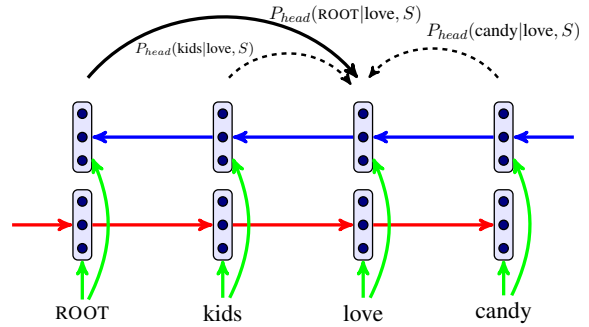
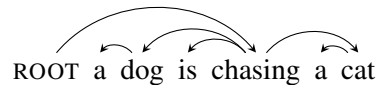


Figure 1: DENSE estimates the probability a word being the head of another word based on bidirectional LSTM representations for the two words. $P_{head}(\text{ROOT}|\text{love}, S)$ is the probability of ROOT being the head of *love* (dotted arcs denote candidate heads; the solid arc is the goldstandard).

mation about each w_i in isolation (e.g., its lexical meaning or POS tag), \mathbf{a}_i must also encode w_i 's positional information within the sentence. Such information has been shown to be important in dependency parsing (McDonald et al., 2005a). For example, in the following sentence:



the head of the first *a* is *dog*, whereas the head of the second *a* is *cat*. Without considering positional information, a model cannot easily decide which *a* (nearer or farther) to assign to *dog*.

Long short-term memory networks (Hochreiter and Schmidhuber, 1997; LSTMs), a type of recurrent neural network with a more complex computational unit, have proven effective at capturing long-term dependencies. In our case LSTMs allow to represent each word on its own and within a sequence leveraging long-range contextual information. As shown in Figure 1, we first use a forward LSTM (LSTM^F) to read the sentence from left to right and then a backward LSTM (LSTM^B) to read the sentence from right to left, so that the entire sentence serves as context for each word:²

$$\mathbf{h}_i^F, \mathbf{c}_i^F = \text{LSTM}^F(\mathbf{x}_i, \mathbf{h}_{i-1}^F, \mathbf{c}_{i-1}^F) \quad (1)$$

$$\mathbf{h}_i^B, \mathbf{c}_i^B = \text{LSTM}^B(\mathbf{x}_i, \mathbf{h}_{i+1}^B, \mathbf{c}_{i+1}^B) \quad (2)$$

where \mathbf{x}_i is the feature vector of word w_i , $\mathbf{h}_i^F \in \mathbb{R}^d$ and $\mathbf{c}_i^F \in \mathbb{R}^d$ are the hidden states and memory cells for the i th word w_i in LSTM^F and d is

²For more detail on LSTM networks, see e.g., Graves (2012) or Goldberg (2016).

the hidden unit size. \mathbf{h}_i^F is also the representation for $w_{0:i}$ (w_i and its left neighboring words) and \mathbf{c}_i^F is an internal state maintained by LSTM^F. $\mathbf{h}_i^B \in \mathbb{R}^d$ and $\mathbf{c}_i^B \in \mathbb{R}^d$ are the hidden states and memory cells for the backward LSTM^B. Each token w_i is represented by \mathbf{x}_i , the concatenation of two vectors corresponding to w_i 's lexical and POS tag embeddings:

$$\mathbf{x}_i = [\mathbf{W}_e \cdot e(w_i); \mathbf{W}_t \cdot e(t_i)] \quad (3)$$

where $e(w_i)$ and $e(t_i)$ are one-hot vector representations of token w_i and its POS tag t_i ; $\mathbf{W}_e \in \mathbb{R}^{s \times |V|}$ and $\mathbf{W}_t \in \mathbb{R}^{q \times |T|}$ are the word and POS tag embedding matrices, where $|V|$ is the vocabulary size, s is the word embedding size, $|T|$ is the POS tag set size, and q the tag embedding size. The hidden states of the forward and backward LSTMs are concatenated to obtain \mathbf{a}_i , the final representation of w_i :

$$\mathbf{a}_i = [\mathbf{h}_i^F; \mathbf{h}_i^B] \quad i \in [0, N] \quad (4)$$

Note that bidirectional LSTMs are one of many possible ways of representing word w_i . Alternative representations include embeddings obtained from feed-forward neural networks (Chen and Manning, 2014; Lei et al., 2014a), character-based embeddings (Ballesteros et al., 2015), and more conventional features such as those introduced in McDonald et al. (2005a).

3.2 Head Selection

We now move on to discuss our formalization of dependency parsing as head selection. We begin with unlabeled dependencies and then explain how the model can be extended to predict labeled ones.

In a dependency tree, a head can have multiple dependents, whereas a dependent can have only one head. Based on this fact, dependency parsing can be formalized as follows. Given a sentence $S = (w_0, w_1, \dots, w_N)$, we aim to find for each word $w_i \in \{w_1, w_2, \dots, w_n\}$ the most probable head $w_j \in \{w_0, w_1, \dots, w_N\}$. For example, in Figure 1, to find the head for the token *love*, we calculate probabilities $P_{head}(\text{ROOT}|\text{love}, S)$, $P_{head}(\text{kids}|\text{love}, S)$, and $P_{head}(\text{candy}|\text{love}, S)$, and select the highest. More formally, we estimate the probability of token w_j being the head of token w_i in sentence S as:

$$P_{head}(w_j|w_i, S) = \frac{\exp(g(\mathbf{a}_j, \mathbf{a}_i))}{\sum_{k=0}^N \exp(g(\mathbf{a}_k, \mathbf{a}_i))} \quad (5)$$

where \mathbf{a}_i and \mathbf{a}_j are vector-based representations of w_i and w_j , respectively (described in Section 3.1); $g(\mathbf{a}_j, \mathbf{a}_i)$ is a neural network with a single hidden layer that computes the associative score between representations \mathbf{a}_i and \mathbf{a}_j :

$$g(\mathbf{a}_j, \mathbf{a}_i) = \mathbf{v}_a^\top \cdot \tanh(\mathbf{U}_a \cdot \mathbf{a}_j + \mathbf{W}_a \cdot \mathbf{a}_i) \quad (6)$$

where $\mathbf{v}_a \in \mathbb{R}^{2d}$, $\mathbf{U}_a \in \mathbb{R}^{2d \times 2d}$, and $\mathbf{W}_a \in \mathbb{R}^{2d \times 2d}$ are weight matrices of g . Note that the candidate head w_j can be the ROOT, while the dependent w_i cannot. Equations (5) and (6) compute the probability of adding an arc between two words, in a fashion similar to the neural attention mechanism in sequence-to-sequence models (Bahdanau et al., 2015).

We train our model by minimizing the negative log likelihood of the gold standard $\langle \text{head}, \text{dependent} \rangle$ arcs in all training sentences:

$$J(\theta) = -\frac{1}{|\mathcal{T}|} \sum_{S \in \mathcal{T}} \sum_{i=1}^{N_S} \log P_{head}(h(w_i)|w_i, S) \quad (7)$$

where \mathcal{T} is the training set, $h(w_i)$ is w_i 's gold standard head³ within sentence S , and N_S the number of words in S (excluding ROOT). During inference, for each word w_i ($i \in [1, N_S]$) in S , we greedily choose the most likely head w_j ($j \in [0, N_S]$):

$$w_j = \arg \max_{w_j: j \in [0, N_S]} P_{head}(w_j|w_i, S) \quad (8)$$

Note that the prediction for each word w_i is made independently of the other words in the sentence.

Given our greedy inference method, there is no guarantee that predicted $\langle \text{head}, \text{dependent} \rangle$ arcs form a tree (maybe there are cycles). However, we empirically observed that most outputs during inference are indeed trees. For instance, on an English dataset, 95% of the arcs predicted on the development set are trees, and 87% of them are projective, whereas on a Chinese dataset, 87% of the arcs form trees, 73% of which are projective. This indicates that although the model does not explicitly model tree structure during training, it is able to figure out from the data (which consists of trees) that it should predict them.

So far we have focused on unlabeled dependencies, however it is relatively straightforward to extend DENSE to produce labeled dependencies. We basically train an additional classifier

³Note that $h(w_i)$ can be ROOT.

to predict labels for the arcs which have been already identified. The classifier takes as input features $[\mathbf{a}_i; \mathbf{a}_j; \mathbf{x}_i; \mathbf{x}_j]$ representing properties of the arc $\langle w_j, w_i \rangle$. These consist of \mathbf{a}_i and \mathbf{a}_j , the LSTM-based representations for w_i and w_j (see Equation (4)), and their word and part-of-speech embeddings, \mathbf{x}_i and \mathbf{x}_j (see Equation (3)). Specifically, we use a trained DENSE model to go through the training corpus and generate features and corresponding dependency labels as training data. We employ a two-layer rectifier network (Glorot et al., 2011) for the classification task.

3.3 Maximum Spanning Tree Algorithms

As mentioned earlier, greedy inference may not produce well-formed trees. In this case, the output of DENSE can be adjusted with a maximum spanning tree algorithm. We use the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) for building non-projective trees and the Eisner (1996) algorithm for projective ones.

Following McDonald et al. (2005b), we view a sentence $S = (w_0 = \text{ROOT}, w_1, \dots, w_N)$ as a graph $G_S = \langle V_S, E_S \rangle$ with the sentence words and the dummy root symbol as vertices and a directed edge between every pair of distinct words and from the root symbol to every word. The directed graph G_S is defined as:

$$\begin{aligned} V_S &= \{w_0 = \text{ROOT}, w_1, \dots, w_N\} \\ E_S &= \{\langle i, j \rangle : i \neq j, \langle i, j \rangle \in [0, N] \times [1, N]\} \\ s(i, j) &= P_{\text{head}}(w_i | w_j, S) \quad \langle i, j \rangle \in E_S \end{aligned}$$

where $s(i, j)$ is the weight of edge $\langle i, j \rangle$ and $P_{\text{head}}(w_i | w_j, S)$ is known. The problem of dependency parsing now boils down to finding the tree with the highest score which is equivalent to finding a MST in G_S (McDonald et al., 2005b).

Non-projective Parsing To build a non-projective parser, we solve the MST problem with the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). The algorithm selects for each vertex (excluding ROOT) the in-coming edge with the highest weight. If a tree results, it must be the maximum spanning tree and the algorithm terminates. Otherwise, there must be a cycle which the algorithm identifies, contracts into a single vertex and recalculates edge weights going into and out of the cycle. The greedy inference strategy described in Equation (8)) is essentially a sub-procedure in the Chu-Liu-Edmonds algorithm

with the algorithm terminating after the first iteration. In implementation, we only run the Chu-Liu-Edmonds algorithm through graphs with cycles, i.e., non-tree outputs.

Projective Parsing For projective parsing, we solve the MST problem with the Eisner (1996) algorithm. The time complexity of the Eisner algorithm is $O(N^3)$, while checking if a tree is projective can be done reasonably faster, with a $O(N \log N)$ algorithm. Therefore, we only apply the Eisner algorithm to the non-projective output of our greedy inference strategy. Finally, it should be noted that the *training* of our model does not rely on the Chu-Liu-Edmonds or Eisner algorithm, or any other graph-based algorithm. MST algorithms are only used at *test* time to correct non-tree outputs which are a minority; DENSE acquires underlying tree structure constraints from the data without an explicit learning algorithm.

4 Experiments

We evaluated our parser in a projective and non-projective setting. In the following, we describe the datasets we used and provide training details for our models. We also present comparisons against multiple previous systems and analyze the parser’s output.

4.1 Datasets

In the projective setting, we assessed the performance of our parser on the English Penn Treebank (PTB) and the Chinese Treebank 5.1 (CTB). Our experimental setup closely follows Chen and Manning (2014) and Dyer et al. (2015).

For English, we adopted the Stanford basic dependencies (SD) representation (De Marneffe et al., 2006).⁴ We follow the standard splits of PTB, sections 2–21 were used for training, section 22 for development, and section 23 for testing. POS tags were assigned using the Stanford tagger (Toutanova et al., 2003) with an accuracy of 97.3%. For Chinese, we follow the same split of CTB5 introduced in Zhang and Clark (2008). In particular, we used sections 001–815, 1001–1136 for training, sections 886–931, 1148–1151 for development, and sections 816–885, 1137–1147 for testing. The original constituency trees in CTB were converted to dependency trees with the

⁴We obtained SD representations using the Stanford parser v.3.3.0.

Dataset	# Sentences	(%) Projective
English	39,832	99.9
Chinese	16,091	100.0
Czech	72,319	76.9
German	38,845	72.2

Table 1: Projective statistics on four datasets. Number of sentences and percentage of projective trees are calculated on the training set.

Penn2Malt tool.⁵ We used gold segmentation and gold POS tags as in Chen and Manning (2014) and Dyer et al. (2015).

In the non-projective setting, we assessed the performance of our parser on Czech and German, the largest non-projective datasets released as part of the CoNLL 2006 multilingual dependency parsing shared task. Since there is no official development set in either dataset, we used the last 374/367 sentences in the Czech/German training set as development data.⁶ Projective statistics of the four datasets are summarized in Table 1.

4.2 Training Details

We trained our models on an Nvidia GPU card; training takes one to two hours. Model parameters were uniformly initialized to $[-0.1, 0.1]$. We used Adam (Kingma and Ba, 2014) to optimize our models with hyper-parameters recommended by the authors (i.e., learning rate 0.001, first momentum coefficient 0.9, and second momentum coefficient 0.999). To alleviate the gradient exploding problem, we rescaled the gradient when its norm exceeded 5 (Pascanu et al., 2013). Dropout (Srivastava et al., 2014) was applied to our model with the strategy recommended in the literature (Zaremba et al., 2014; Semeniuta et al., 2016). On all datasets, we used two-layer LSTMs and set $d = s = 300$, where d is the hidden unit size and s is the word embedding size.

As in previous neural dependency parsing work (Chen and Manning, 2014; Dyer et al., 2015), we used pre-trained word vectors to initialize our word embedding matrix \mathbf{W}_e . For the PTB experiments, we used 300 dimensional pre-trained GloVe⁷ vectors (Pennington et al., 2014). For the CTB experiments, we trained 300 dimensional

⁵<http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html>

⁶We make the number of sentences in the development and test sets comparable.

⁷<http://nlp.stanford.edu/projects/glove/>

Parser	Dev		Test	
	UAS	LAS	UAS	LAS
Bohnet10	—	—	92.88	90.71
Martins13	—	—	92.89	90.55
Z&M14	—	—	93.22	91.02
Z&N11	—	—	93.00	90.95
C&M14	92.00	89.70	91.80	89.60
Dyer15	93.20	90.90	93.10	90.90
Weiss15	—	—	93.99	92.05
Andor16	—	—	94.61	92.79
K&G16 <i>graph</i>	—	—	93.10	91.00
K&G16 <i>trans</i>	—	—	93.90	91.90
DENSE-Pei	90.77	88.35	90.39	88.05
DENSE-Pei+E	91.39	88.94	91.00	88.61
DENSE	94.17	91.82	94.02	91.84
DENSE+E	94.30	91.95	94.10	91.90

Table 2: Results on English dataset (PTB with Stanford Dependencies). +E: we post-process non-projective output with the Eisner algorithm.

GloVe vectors on the Chinese Gigaword corpus which we segmented with the Stanford Chinese Segmenter (Tseng et al., 2005). For Czech and German, we did not use pre-trained word vectors. The POS tag embedding size was set to $q = 30$ in the English experiments, $q = 50$ in the Chinese experiments and $q = 40$ in both Czech and German experiments.

4.3 Results

For both English and Chinese experiments, we report unlabeled (UAS) and labeled attachment scores (LAS) on the development and test sets; following Chen and Manning (2014) punctuation is excluded from the evaluation.

Experimental results on PTB are shown in Table 2. We compared our model with several recent papers following the same evaluation protocol and experimental settings. The first block in the table contains mostly graph-based parsers which do not use neural networks: Bohnet10 (Bohnet, 2010), Martins13 (Martins et al., 2013), and Z&M14 (Zhang and McDonald, 2014). Z&N11 (Zhang and Nivre, 2011) is a transition-based parser with non-local features. Accuracy results for all four parsers are reported in Weiss et al. (2015).

The second block in Table 2 presents results obtained from neural network-based parsers. C&M14 (Chen and Manning, 2014) is a transition-based parser using features learned with a feed

Parser	Dev		Test	
	UAS	LAS	UAS	LAS
Z&N11	—	—	86.00	84.40
Z&M14	—	—	87.96	86.34
C&M14	84.00	82.40	83.90	82.40
Dyer15	87.20	85.90	87.20	85.70
K&G16 <i>graph</i>	—	—	86.60	85.10
K&G16 <i>trans</i>	—	—	87.60	86.10
DENSE-Pei	82.50	80.74	82.38	80.55
DENSE-Pei+E	83.40	81.63	83.46	81.65
DENSE	87.27	85.73	87.63	85.94
DENSE+E	87.35	85.85	87.84	86.15

Table 3: Results on Chinese dataset (CTB). +E: we post-process non-projective outputs with the Eisner algorithm.

Parser	PTB		CTB	
	Dev	Test	Dev	Test
C&M14	43.35	40.93	32.75	32.20
Dyer15	51.94	50.70	39.72	37.23
DENSE	51.24	49.34	34.74	33.66
DENSE+E	52.47	50.79	36.49	35.13

Table 4: UEM results on PTB and CTB.

forward neural network. Although very fast, its performance is inferior compared to graph-based parsers or strong non-neural transition based parsers (e.g., Z&N11). Dyer15 (Dyer et al., 2015) uses (stack) LSTMs to model the states of the buffer, the stack, and the action sequence of a transition system. Weiss15 (Weiss et al., 2015) is another transition-based parser, with a more elaborate training procedure. Features are learned with a neural network model similar to C&M14, but much larger with two layers. The hidden states of the neural network are then used to train a structured perceptron for better beam search decoding. Andor16 (Andor et al., 2016) is similar to Weiss15, but uses a globally normalized training algorithm instead.

Unlike all models above, DENSE does not use any kind of transition- or graph-based algorithm during training and inference. Nonetheless, it obtains a UAS of 94.02%. Around 95% of the model’s outputs after inference are trees, 87% of which are projective. When we post-process the remaining 13% of non-projective outputs with the Eisner algorithm (DENSE+E), we obtain a slight improvement on UAS (94.10%).

Kiperwasser and Goldberg (2016) extract fea-

Parser	Czech		German	
	UAS	LAS	UAS	LAS
MST-1st	86.18	—	89.54	—
MST-2nd	87.30	—	90.14	—
Turbo-1st	87.66	—	90.52	—
Turbo-3rd	90.32	—	92.41	—
RBG-1st	87.90	—	90.24	—
RBG-3rd	90.50	—	91.97	—
DENSE-Pei	86.00	77.92	89.42	86.48
DENSE-Pei+CLE	86.52	78.42	89.52	86.58
DENSE	89.60	81.70	92.15	89.58
DENSE+CLE	89.68	81.72	92.19	89.60

Table 5: Non-projective results on the CoNLL 2006 dataset. +CLE: we post-process non-tree outputs with the Chu-Liu-Edmonds algorithm.

tures from bidirectional LSTMs and feed them to a graph- (K&G16 *graph*) and transition-based parser (K&G16 *trans*). Their LSTMs are jointly trained with the parser objective. DENSE yields very similar performance to their transition-based parser while it outperforms K&G16 *graph*. A key difference between DENSE and K&G16 lies in the training objective. The objective of DENSE is log-likelihood based *without* tree structure constraints (the model is trained to produce a distribution over possible heads for each word, where each head selection is independent), while K&G16 employ a max-margin objective *with* tree structure constraints. Although our probabilistic objective is non-structured, it is perhaps easier to train compared to a margin-based one.

We also assessed the importance of the bidirectional LSTM on its own by replacing our LSTM-based features with those obtained from a feed-forward network. Specifically, we used the 1-order-atomic features introduced in Lei et al. (2014a) which represent POS-tags, modifiers, heads, and their relative positions. As can be seen in Table 2 (DENSE-Pei), these features are less effective compared to LSTM-based ones and the contribution of the MST algorithm (Eisner) during decoding is more pronounced (DENSE-Pei+E). We observe similar trends in the Chinese, German, and Czech datasets (see Tables 3 and 5).

Results on CTB follow a similar pattern. As shown in Table 3, DENSE outperforms all previous neural models (see the test set columns) on UAS and LAS. DENSE performs competitively with Z&M14, a non-neural model with a com-

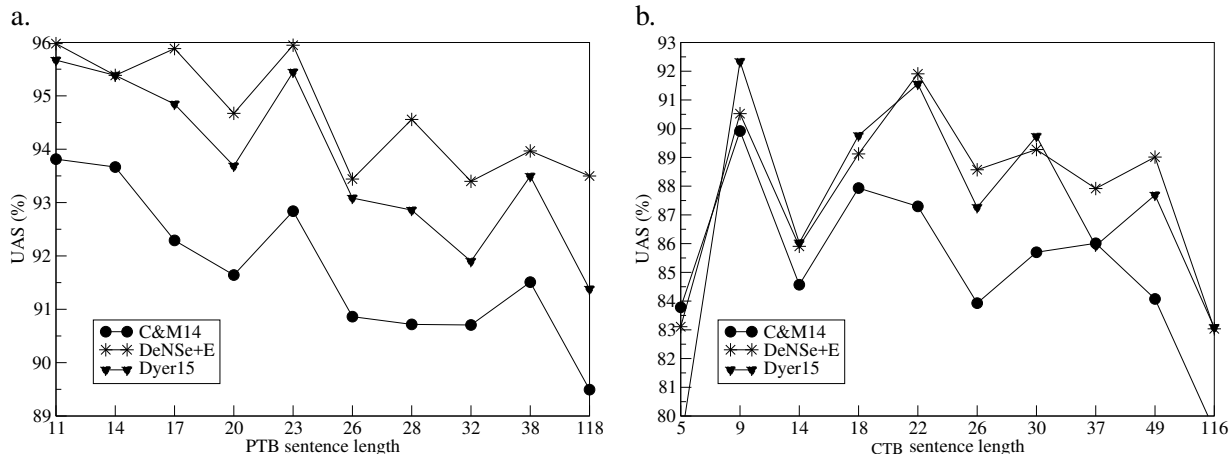


Figure 2: UAS against sentence length on PTB and CTB (development set). Sentences are sorted by length in ascending order and divided equally into 10 bins. The horizontal axis is the length of the last sentence in each bin.

plex high order decoding algorithm involving cube pruning and strategies for encouraging diversity. Post-processing the output of the parser with the Eisner algorithm generally improves performance (by 0.21%; see last row in Table 3). Again we observe that 1-order-atomic features (Lei et al., 2014a) are inferior compared to the LSTM. Table 4 reports unlabeled sentence level exact match (UEM) in Table 4 for English and Chinese. Interestingly, even when using the greedy inference strategy, DENSE yields a UEM comparable to Dyer15 on PTB. Finally, in Figure 2 we analyze the performance of our parser on sentences of different length. On both PTB and CTB, DENSE has an advantage on long sentences compared to C&M14 and Dyer15.

For Czech and German, we closely follow the evaluation setup of CoNLL 2006. We report both UAS and LAS, although most previous work has focused on UAS. Our results are summarized in Table 5. We compare DENSE against three non-projective graph-based dependency parsers: the MST parser (McDonald et al., 2005b), the Turbo parser (Martins et al., 2013), and the RBG parser (Lei et al., 2014b). We show the performance of these parsers in the first order setting (e.g., MST-1st) and in higher order settings (e.g., Turbo-3rd). The results of MST-1st, MST-2nd, RBG-1st and RBG-3rd are reported in Lei et al. (2014b) and the results of Turbo-1st and Turbo-3rd are reported in Martins et al. (2013). We show results for our parser with greedy inference (see DENSE in the table) and when we use the Chu-

Dataset	#Sent	Before MST		After MST	
		Tree	Proj	Tree	Proj
PTB	1,700	95.1	86.6	100.0	100.0
CTB	803	87.0	73.1	100.0	100.0
Czech	374	87.7	65.5	100.0	72.7
German	367	96.7	67.3	100.0	68.1

Table 6: Percentage of trees and projective trees on the development set before and after DENSE uses a MST algorithm. On PTB and CTB, we use the Eisner algorithm and on Czech and German, we use the Chu-Liu-Edmonds algorithm.

Liu-Edmonds algorithm to post-process non-tree outputs (DENSE+CLE).

As can be seen, DENSE outperforms all other first (and second) order parsers on both German and Czech. As in the projective experiments, we observe slight a improvement (on both UAS and LAS) when using a MST algorithm. On German, DENSE is comparable with the best third-order parser (Turbo-3rd), while on Czech it lags behind Turbo-3rd and RBG-3rd. This is not surprising considering that DENSE is a first-order parser and only uses words and POS tags as features. Comparison systems use a plethora of hand-crafted features and more sophisticated high-order decoding algorithms. Finally, note that a version of DENSE with features in (Lei et al., 2014a) is consistently worse (see the second block in Table 5).

Our experimental results demonstrate that using a MST algorithm during inference can slightly improve the model’s performance. We further ex-

amined the extent to which the MST algorithm is necessary for producing dependency trees. Table 6 shows the percentage of trees before and after the application of the MST algorithm across the four languages. In the majority of cases DENSE outputs trees (ranging from 87.0% to 96.7%) and a significant proportion of them are projective (ranging from 65.5% to 86.6%). Therefore, only a small proportion of outputs (14.0% on average) need to be post-processed with the Eisner or Chu-Liu-Edmonds algorithm.

5 Conclusions

In this work we presented DENSE, a neural dependency parser which we train without a transition system or graph-based algorithm. Experimental results show that DENSE achieves competitive performance across four different languages and can seamlessly transfer from a projective to a non-projective parser simply by changing the post-processing MST algorithm during inference. In the future, we plan to increase the coverage of our parser by using tri-training techniques (Li et al., 2014) and multi-task learning (Luong et al., 2015).

Acknowledgments We would like to thank Adam Lopez and Frank Keller for their valuable feedback. We acknowledge the financial support of the European Research Council (ERC; award number 681760).

References

- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California.
- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal.
- Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China.
- Xavier Carreras and Michael Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209, Singapore.
- Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic.
- Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. 1997. Structure and performance of a dependency language model. In *Fifth European Conference on Speech Communication and Technology, EUROSPEECH 1997*, Rhodes, Greece.
- Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.
- Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8.
- Koby Crammer and Yoram Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.
- Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.
- James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany.
- Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, Genoa, Italy.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual*

- Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 334–343, Beijing, China.
- Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.
- Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86, Santa Cruz, California, USA.
- Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.
- Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.
- Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, Cadiz, Spain.
- Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India.
- Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.
- Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer.
- Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 392–399, Prague, Czech Republic.
- James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL’04), Main Volume*, pages 95–102, Barcelona, Spain.
- Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.
- Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA.
- Sandra Kübler, Ryan McDonald, Joakim Nivre, and Graeme Hirst. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.
- Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014a. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland.
- Tao Lei, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014b. Low-rank tensors for scoring dependency structures. In *Proceedings of the ACL*. Baltimore, Maryland.
- Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 457–467, Baltimore, Maryland.
- Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.
- Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria.
- Marshall R. Mayberry and Risto Miikkulainen. 1999. SardSn: A neural network shift-reduce parser. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 820–825, Stockholm, Sweden.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98, Ann Arbor, Michigan.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural*

- Language Processing*, pages 523–530, Vancouver, British Columbia, Canada.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2006a. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, Genoa, Italy.
- Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City.
- Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–160, Nancy, France.
- Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain.
- Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, Atlanta, Georgia.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.
- Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2016. Recurrent dropout without memory loss. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1757–1766, Osaka, Japan.
- David Arthur Smith. 2010. *Efficient inference for trees and alignments: modeling monolingual and bilingual syntax with hard and soft constraints and latent variables*. Johns Hopkins University.
- Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304, Vancouver, British Columbia.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- Ivan Titov and James Henderson. 2007. Constituent parsing with incremental sigmoid belief networks. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 632–639, Prague, Czech Republic.
- Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 173–180, Edmonton, Canada.
- Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for Sighan bake-off 2005. In *Proceedings of the 4th SIGHAN workshop on Chinese language Processing*, pages 168–171, Jeju Island, Korea.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th Workshop on Parsing Technologies*, pages 195–206, Nancy, France.
- Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.
- Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii.
- Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331, Jeju Island, Korea.
- Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 656–661, Baltimore, Maryland.
- Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016.
Top-down tree long short-term memory networks.
In *Proceedings of the 2016 Conference of the North
American Chapter of the Association for Computa-
tional Linguistics: Human Language Technologies*,
pages 310–320, San Diego, California.

Tackling Error Propagation through Reinforcement Learning: A Case of Greedy Dependency Parsing

Minh Lê
CLTL

Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.n.le@vu.nl

Antske Fokkens
CLTL

Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
antske.fokkens@vu.nl

Abstract

Error propagation is a common problem in NLP. Reinforcement learning explores erroneous states during training and can therefore be more robust when mistakes are made early in a process. In this paper, we apply reinforcement learning to greedy dependency parsing which is known to suffer from error propagation. Reinforcement learning improves accuracy of both labeled and unlabeled dependencies of the Stanford Neural Dependency Parser, a high performance greedy parser, while maintaining its efficiency. We investigate the portion of errors which are the result of error propagation and confirm that reinforcement learning reduces the occurrence of error propagation.

1 Introduction

Error propagation is a common problem for many NLP tasks (Song et al., 2012; Quirk and Corston-Oliver, 2006; Han et al., 2013; Gildea and Palmer, 2002; Yang and Cardie, 2013). It can occur when NLP tools applied early on in a pipeline make mistakes that have negative impact on higher-level tasks further down the pipeline. It can also occur within the application of a specific task, when sequential decisions are taken and errors made early in the process affect decisions made later on.

When reinforcement learning is applied, a system actively tries out different sequences of actions. Most of these sequences will contain some errors. We hypothesize that a system trained in this manner will be more robust and less susceptible to error propagation.

We test our hypothesis by applying reinforcement learning to greedy transition-based parsers (Yamada and Matsumoto, 2003; Nivre, 2004),

which have been popular because of superior efficiency and accuracy nearing state-of-the-art. They are also known to suffer from error propagation. Because they work by carrying out a sequence of actions without reconsideration, an erroneous action can exert a negative effect on all subsequent decisions. By rendering correct parses unreachable or promoting incorrect features, the first error induces the second error and so on. McDonald and Nivre (2007) argue that the observed negative correlation between parsing accuracy and sentence length indicates error propagation is at work.

We compare reinforcement learning to supervised learning on Chen and Manning (2014)'s parser. This high performance parser is available as open source. It does not make use of alternative strategies for tackling error propagation and thus provides a clean experimental setup to test our hypothesis. Reinforcement learning increased both unlabeled and labeled accuracy on the Penn TreeBank and German part of SPMRL (Seddah et al., 2014). This outcome shows that reinforcement learning has a positive effect, but does not yet prove that this is indeed the result of reduced error propagation. We therefore designed an experiment which identified which errors are the result of error propagation. We found that around 50% of avoided errors were cases of error propagation in our best arc-standard system. Considering that 27% of the original errors were caused by error propagation, this result confirms our hypothesis.

This paper provides the following contributions:

1. We introduce Approximate Policy Gradient (APG), a new algorithm that is suited for dependency parsing and other structured prediction problems.
2. We show that this algorithm improves the accuracy of a high-performance greedy parser.

3. We design an experiment for analyzing error propagation in parsing.
4. We confirm our hypothesis that reinforcement learning reduces error propagation.

To our knowledge, this paper is the first to experimentally show that reinforcement learning can reduce error propagation in NLP.

The rest of this paper is structured as follows. We discuss related work in Section 2. This is followed by a description of the parsers used in our experiments in Section 3. Section 4 outlines our experimental setup and presents our results. The error propagation experiment and its outcome are described in Section 5. Finally, we conclude and discuss future research in Section 6.

2 Related Work

In this section, we address related work on dependency parsing, including alternative approaches for reducing error propagation, and reinforcement learning.

2.1 Dependency Parsing

We use Chen and Manning (2014)’s parser as a basis for our experiments. Their parser is open-source and has served as a reference point for many recent publications (Dyer et al., 2015; Weiss et al., 2015; Alberti et al., 2015; Honnibal and Johnson, 2015, among others). They provide an efficient neural network that learns dense vector representations of words, PoS-tags and dependency labels. This small set of features makes their parser significantly more efficient than other popular parsers, such as the Malt (Nivre et al., 2007) or MST (McDonald et al., 2005) parser while obtaining higher accuracy. They acknowledge the error propagation problem of greedy parsers, but leave addressing this through (e.g.) beam search for future work.

Dyer et al. (2015) introduce an approach that uses Long Short-Term Memory (LSTM). Their parser still works incrementally and the number of required operations grows linearly with the length of the sentence, but it uses the complete buffer, stack and history of parsing decisions, giving the model access to global information. Weiss et al. (2015) introduce several improvements on Chen and Manning (2014)’s parser. Most importantly, they put a globally-trained perceptron layer instead of a softmax output layer. Their model uses

smaller embeddings, rectified linear instead of cubic activation function, and two hidden layers instead of one. They furthermore apply an averaged stochastic gradient descent (ASGD) learning scheme. In addition, they apply beam search and increase training data by using unlabeled data through the tri-training approach introduced by Li et al. (2014), which leads to further improvements.

Kiperwasser and Goldberg (2016) introduce a new way to represent features using a bidirectional LSTM and improve the results of a greedy parser. Andor et al. (2016) present a mathematical proof that globally normalized models are more expressive than locally normalized counterparts and propose to use global normalization with beam search at both training and testing.

Our approach differs from all of the work mentioned above, in that it manages to improve results of Chen and Manning (2014) *without changing the architecture of the model nor the input representation*. The only substantial difference lies in the way the model is trained. In this respect, our research is most similar to training approaches using dynamic oracles (Goldberg and Nivre, 2012). Traditional static oracles can generate only one sequence of actions per sentence. A dynamic oracle gives *all* trajectories leading to the best possible result from every valid parse configuration. They can therefore be used to generate more training sequences including those containing errors. A drawback of this approach is that dynamic oracles have to be developed specifically for individual transition systems (e.g. arc-standard, arc-eager). Therefore, a large number of dynamic oracles have been developed in recent years (Goldberg and Nivre, 2012; Goldberg and Nivre, 2013; Goldberg et al., 2014; Gomez-Rodriguez et al., 2014; Björkelund and Nivre, 2015). In contrast, the reinforcement learning approach proposed in this paper is more general and can be applied to a variety of systems.

Zhang and Chan (2009) present the only study we are aware of that also uses reinforcement learning for dependency parsing. They compare their results to Nivre et al. (2006b) using the same features, but they also change the model and apply beam search. It is thus unclear to what extent their improvements are due to reinforcement learning.

Even though most approaches mentioned above improve the results reported by Chen and Manning (2014) and even more impressive results on

dependency parsing have been achieved since (notably, Andor et al. (2016)), Chen and Manning’s parser provides a better baseline for our purposes. We aim at investigating the influence of reinforcement learning on error propagation and want to test this in a clean environment, where reinforcement learning does not interfere with other methods that address the same problem.

2.2 Reinforcement Learning

Reinforcement learning has been applied to several NLP tasks with success, e.g. agenda-based parsing (Jiang et al., 2012), semantic parsing (Berant and Liang, 2015) and simultaneous machine translation (Grissom II et al., 2014). To our knowledge, however, none of these studies investigated the influence of reinforcement learning on error propagation.

Learning to Search (L2S) is probably the most prominent line of research that applies reinforcement learning (more precisely, imitation learning) to NLP. Various algorithms, e.g. SEARN (Daumé III et al., 2009) and DAGger (Ross et al., 2011), have been developed sharing common high-level steps: a *roll-in* policy is executed to generate training states from which a *roll-out* policy is used to estimate the loss of certain actions. The concrete instantiation differs from one algorithm to another with choices including a referent policy (static or dynamic oracle), learned policy, or a mixture of the two. Early work in L2S focused on reducing reinforcement learning into binary classification (Daumé III et al., 2009), but newer systems favored regressors for efficiency (Chang et al., 2015, Supplementary material, Section B). Our algorithm APG is simpler than L2S in that it uses only one policy (pre-trained with standard supervised learning) and applies the existing classifier directly without reduction (the only requirement is that it is probabilistic). Nevertheless, our results demonstrate its effectiveness.

APG belongs to the family of policy gradient algorithms (Sutton et al., 1999), i.e. it maximizes the expected reward directly by following its gradient w.r.t. the parameters. The advantage of using a policy gradient algorithm in NLP is that gradient-based optimization is already widely used. REINFORCE (Williams, 1992; Ranzato et al., 2016) is a widely-used policy gradient algorithm but it is also well-known for suffering from high variance (Sutton et al., 1999).

We directly compare our approach to REINFORCE, whereas we leave a direct comparison to L2S for future work. Our experiments show that our algorithm results in lower variance and achieves better performance than REINFORCE.

Recent work addresses the approximation of reinforcement learning gradient in the context of machine translation. Shen et al. (2016)’s algorithm is roughly equivalent to the combination of an oracle and random sampling. Their approach differs from ours, because it does not retain memory across iteration as in our best-performing model (see Section 3.4).

2.3 Reinforcement and error propagation

As mentioned above, previous work that applied reinforcement learning to NLP has, to our knowledge, not shown that it improved results by reducing error propagation.

Work on identifying the impact of error propagation in parsing is rare, Ng and Curran (2015) being a notable exception. They provide a detailed error analysis for parsing and classify which kind of parsing errors are involved with error propagation. There are four main differences between their approaches and ours. First, Ng and Curran correct arcs in the tree and our algorithm corrects decisions of the parsing algorithm. Second, our approach distinguishes between cases where one erroneous action deterministically leads to multiple erroneous arcs and cases where an erroneous action leads to conditions that indirectly result in further errors (see Section 5.1 for a detailed explanation). Third, Ng and Curran’s algorithm corrects all erroneous arcs that are the same type of parsing error and point out that they cannot examine the interaction between multiple errors of the same type in a sentence. Our algorithm corrects errors incrementally and therefore avoids this issue. Finally, the classification and analysis presented in Ng and Curran (2015) are more extensive and detailed than ours. Our algorithm can, however, easily be extended to perform similar analysis. Overall, Ng and Curran’s approach for error analysis and ours are complementary. Combining them and applying them to various systems would form an interesting direction for future work.

3 A Reinforced Greedy Parser

This section describes the systems used in our experiments. We first describe the arc-standard al-

Step	Transition	Stack	Buffer	Arcs
0		<ROOT>	waves hit ... Big Board	\emptyset
1	SHIFT	<ROOT> waves	hit stocks ... Big Board	\emptyset
2	SHIFT	<ROOT> waves hit	stocks themselves ... Big Board	\emptyset
3	LEFT _{nsubj}	<ROOT> hit	stocks themselves ... Big Board	$A_1 = \{ hit \xrightarrow{nsubj} waves \}$
4	SHIFT	<ROOT> hit stocks	themselves on the Big Board	A_1
5	SHIFT	<ROOT> hit stocks themselves	on the Big Board	A_1
6	RIGHT _{dep}	<ROOT> hit stocks	on the Big Board	$A_2 = A_1 \cup \{ stock \xrightarrow{dep} themselves \}$
7	RIGHT _{doobj}	<ROOT> hit	on the Big Board	$A_3 = A_2 \cup \{ hit \xrightarrow{doobj} stock \}$

Table 1: Parsing oracle walk-through

gorithm, because familiarity with it helps to understand our error propagation analysis. Next, we briefly point out the main differences between the arc-standard algorithm and the alternative algorithms we experimented with (arc-eager and swap-standard). We then outline the traditional and our novel machine learning approaches. The features we used are identical to those described in Chen and Manning (2014). We are not aware of research identifying the best feature for a neural parser with arc-eager or swap-standard so we use the same features for all transition systems.

3.1 Transition-Based Dependency Parsing

In an arc-standard system (Nivre, 2004), a parsing configuration consists of a triple $\langle \Sigma, \beta, A \rangle$, where Σ is a stack, β is a buffer containing the remaining input tokens and A are the dependency arcs that are created during parsing process. At initiation, the stack contains only the root symbol ($\Sigma = [\text{ROOT}]$), the buffer contains the tokens of the sentence ($\beta = [w_1, \dots, w_n]$) and the set of arcs is empty ($A = \emptyset$).

The arc-standard system supports three transitions. When σ_1 is the top element and σ_2 the second element on the stack, and β_1 the first element of the buffer:¹

LEFT_l adds an arc $\sigma_1 \xrightarrow{l} \sigma_2$ to A and removes σ_2 from the stack.

RIGHT_l adds an arc $\sigma_2 \xrightarrow{l} \sigma_1$ to A and removes σ_1 from the stack.

SHIFT moves β_1 to the stack.

When the buffer is empty, the stack contains only the root symbol and A contains a parse tree, the configuration is completed. For a sentence of

¹Naturally, the transitions LEFT_l and RIGHT_l can only take place if the stack contains at least two elements and SHIFT can only occur when there is at least one element on the buffer.

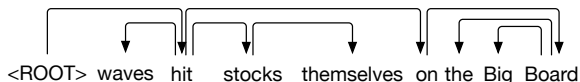


Figure 1: Correct dependencies for a simplified example from Penn TreeBank

N_w tokens, a full parse takes $2N_w + 1$ transitions to complete (including the initiation). Figure 1 provides the gold parse tree for a (simplified) example from the Penn Treebank. The steps taken to create the dependencies between the sentence’s head word *hit* and its subject and direct object are provided in Table 1.

To demonstrate that reinforcement learning can train different systems, we also carried out experiments with arc-eager (Nivre, 2003) and swap-standard (Nivre, 2009). Arc-eager is designed for incremental parsing and included in the popular MaltParser (Nivre et al., 2006a). Swap-standard is a simple and effective solution to unprojective dependency trees. Because arc-eager does not guarantee complete parse trees, we used a variation that employs an action called UNSHIFT to resume processing of tokens that would otherwise not be attached to a head (Nivre and Fernández-González, 2014).

3.2 Training with a Static Oracle

In transition-based dependency parsing, it is common to convert a dependency treebank $\mathcal{D} \ni (x, y)$ into a collection of input features $s \in \mathcal{S}$ and corresponding gold-standard actions $a \in \mathcal{A}$ for training, using a static oracle \mathcal{O} . In Chen and Manning (2014), a neural network works as a function mapping input features to probabilities of actions: $f_{NN} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. The neural network is trained to minimize negative log-likelihood loss

on the converted treebank:

$$\mathcal{L} = \sum_{(x,y) \in \mathcal{D}} \sum_{(s,a) \in \mathcal{O}(x,y)} -\log f_{NN}(s, a; \theta) \quad (1)$$

3.3 Reinforcement Learning

Following Maes et al. (2009), we view transition-based dependency parsing as a deterministic Markov Decision Process. The problem is summarized by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r \rangle$ where \mathcal{S} is the set of all possible states, \mathcal{A} contains all possible actions, \mathcal{T} is a mapping $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ called *transition function* and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function.

A state corresponds to a configuration and is summarized into input features. Possible actions are defined for each transition system described in Section 3.1. We keep the training approach simple by using only one reward $r(\bar{y})$ at the end of each parse.

Given this framework, a stochastic policy guides our parser by mapping each state to a probabilistic distribution of actions. During training, we use function f_{NN} described in Section 3.2 as a stochastic policy. At test time, actions are chosen in a greedy fashion following existing literature. We aim at finding the policy that maximizes the expected reward (or, equivalently, minimizes the expected loss) on the training dataset:

$$\text{maximize } \eta = \sum_{(x,y) \in \mathcal{D}} \sum_{a_{1:m} \sim f} r(\bar{y}) \prod_{i=1}^m f_{NN}(s_i, a_i; \theta) \quad (2)$$

where $a_{1:m}$ is a sequence of actions obtained by following policy f_{NN} until termination and $s_{1:m}$ are corresponding states (with s_{m+1} being the termination state).

3.4 Approximate Policy Gradient

Gradient ascent can be used to maximize the expected reward in Equation 2. The gradient of expected reward w.r.t. parameters is (note that $dz = \text{zd}(\log z)$):

$$\frac{\partial \eta}{\partial \theta} = \sum_{(x,y) \in \mathcal{D}} \sum_{a_{1:m} \sim f_{NN}} r(\bar{y}) \prod_{i=1}^m f_{NN}(s_i, a_i) \sum_{i=1}^m \frac{\partial}{\partial \theta} \log f_{NN}(s_i, a_i; \theta) \quad (3)$$

Because of the exponential number of possible trajectories, calculating the gradient exactly is not

possible. We propose to replace it by an approximation (hence the name *Approximate Policy Gradient*) by summing over a small subset U of trajectories. Following common practice, we also use a baseline $b(y)$ that only depends on the correct dependency tree. The parameter is then updated by following the approximate gradient:

$$\Delta \theta \propto \sum_{(x,y) \in \mathcal{D}} \sum_{a_{1:m} \in U} (r(\bar{y}) - b(y)) \prod_{i=1}^m f_{NN}(s_i, a_i) \sum_{i=1}^m \frac{\partial}{\partial \theta} \log f_{NN}(s_i, a_i; \theta) \quad (4)$$

Instead of sampling one trajectory at a time as in REINFORCE, Equation 4 has the advantage that sampling over multiple trajectories could lead to more stable training and higher performance. To achieve that goal, the choice of U is critical. We empirically evaluate three strategies:

RL-ORACLE: only includes the oracle transition sequence.

RL-RANDOM: randomly samples k distinct trajectories at each iteration. Every action is sampled according to f_{NN} , i.e. preferring trajectories for which the current policy assigns higher probability.

RL-MEMORY: samples randomly as the previous method but retains k trajectories with highest rewards across iterations in a separate memory. Trajectories are “forgotten” (removed) randomly with probability ρ before each iteration.²

Intuitively, trajectories that are more likely and produce higher rewards are better training examples. It follows from Equation 3 that they also bear bigger weight on the true gradient. This is the rationale behind RL-RANDOM and RL-ORACLE. For a suboptimal parser, however, these objectives sometimes work against each other. RL-MEMORY was designed to find the right balance between them. It is furthermore important that the parser is pretrained to ensure good samples. Algorithm 1 illustrates the procedure of training a dependency parser using the proposed algorithms.

²We assign a random number (drawn uniformly from $[0, 1]$) to each trajectory in memory and remove those assigned a number less than ρ .

```

MemorySeqs  $\leftarrow$   $\emptyset$ ;
foreach training batch  $b$  do
  foreach sentence  $s \in b$  do
    OracleSeq  $\leftarrow$  Oracle( $s$ );
    SystemSeqs  $\leftarrow$  (sample  $k$  parsing
      transition sequences for  $s$ );
    if RL-Oracle then
      | ComputeGradients(OracleSeq);
    else if RL-Random then
      | ComputeGradients(SystemSeqs);
    else if RL-Memory then
       $m \leftarrow$  MemorySeqs[ $s$ ];
      foreach  $q \in m$  do
        if RandomNumber()  $< \rho$  then
          | Remove  $q$  from  $m$ ;
        end
      end
      foreach  $q \in$  SystemSeqs do
        if  $|m| < k$  then
          | Insert  $q$  into  $m$ ;
        else
           $p \leftarrow$  (sequence with
            smallest reward in  $m$ );
          if reward( $q$ )  $>$  reward( $p$ )
            then
              | Replace  $p$  by  $q$  in  $m$ ;
            end
          end
        end
      ComputeGradients( $m$ );
    end
  Perform one gradient descent step;
end

```

Algorithm 1: Training a dependency parser with approximate policy gradient.

4 Reinforcement Learning Experiments

We first train a parser using a supervised learning procedure and then improve its performance using APG. We empirically tested that training a second time with supervised learning has little to no effect on performance.

4.1 Experimental Setup

We use PENN Treebank 3 with standard split (training, development and test set) for our experiments with arc-standard and arc-eager. Because the swap-standard parser is mainly suited for non-projective structures, which are rare in the PENN Treebank, we evaluate this parser on the German

	Arc-standard		Arc-eager		Swap-standard	
	UAS	LAS	UAS	LAS	UAS	LAS
SL	91.3	89.4	88.3	85.8	84.3	81.3
RE	91.9	90.2	89.7	87.2	87.5	84.4
RL-O	91.8	90.2	88.9	86.5	86.8	83.9
RL-R	92.2	90.6	89.4	87.0	87.5	84.5
RL-M	92.2	90.6	89.8	87.4	87.6	84.6

Table 2: Comparing training methods on PENN Treebank (arc-standard and arc-eager) and German part of SPMRL-2014 (swap-standard).

section of the SPMRL dataset. For PENN Treebank, we follow Chen and Manning’s preprocessing steps. We also use their pretrained model³ for arc-standard and train our own models in similar settings for other transition systems.

For reinforcement learning, we use AdaGrad for optimization. We do not use dropout because we observed that it destabilized the training process. The reward $r(\bar{y})$ is the number of correct labeled arcs (i.e. LAS multiplied by number of tokens).⁴ The baseline is fixed to half the number of tokens (corresponding to a 0.5 LAS score). As training takes a lot of time, we tried only few values of hyperparameters on the development set and picked $k = 8$ and $\rho = 0.01$. 1,000 updates were performed (except for REINFORCE which was trained for 8,000 updates) with each training batch contains 512 randomly selected sentences. The Stanford dependency scorer⁵ was used for evaluation.

4.2 Effectiveness of Reinforcement Learning

Table 2 displays the performance of different approaches to training dependency parsers. Although we used Chen and Manning (2014)’s pretrained model and Stanford open-source software, the results of our baseline are slightly worse than what is reported in their paper. This could be due to minor differences in settings and does not affect our conclusions.

Across transition systems and two languages, APG outperforms supervised learning, verifying our hypothesis. Moreover, it is not simply because the learners are exposed to more examples than their supervised counterparts. RL-ORACLE

³We use PTB_Stanford_params.txt.gz downloaded from <http://nlp.stanford.edu/software/nndep.shtml> on December 30th, 2015.

⁴Punctuation is not taken into account, following Chen and Manning (2014).

⁵Downloaded from <http://nlp.stanford.edu/software/lex-parser.shtml>.

is trained on exactly the same examples as the standard supervised learning system (SL), yet it is consistently superior. This can only be explained by the superiority of the reinforcement learning objective function compared to negative log-likelihood.

The results support our hypothesis that APG is better than REINFORCE (abbreviated as RE in Table 2) as RL-MEMORY always outperforms the classical algorithm and the other two heuristics do in two out of three cases. The usefulness of training examples that contain errors is evident through the better performance of RL-RANDOM and RL-MEMORY in comparison to RL-ORACLE.

Table 3 shows the importance of samples for RL-RANDOM. The algorithm hurts performance when only one sample is used whereas training with two or more samples improves the results. The difference cannot be explained by the total number of observed samples because one-sample training is still worse after 8,000 iterations compared to a sample size of 8 after 1,000 iterations. The benefit of added samples is twofold: increased performance and decreased variance. Because these benefits saturate quickly, we did not test sample sizes beyond 32.

	Dev		Test		Test std.	
	UAS	LAS	UAS	LAS	UAS	LAS
SL	91.5	89.6	91.3	89.4	-	-
RE	92.1*	90.4*	91.9*	90.2*	0.04	0.05
1	91.2*	89.1*	91.0*	88.9*	0.12	0.15
2	91.8*	90.0*	91.6*	89.9*	0.09	0.09
4	92.2*	90.5*	92.0*	90.4*	0.09	0.08
8	92.4*	90.8*	92.2*	90.6*	0.03	0.05
16	92.4	90.8	92.2	90.6	-	-
32	92.4	90.8	92.3	90.6	-	-

Table 3: Parsing accuracy of RL-RANDOM (arc-standard) with different sample sizes compared to supervised learning (SL) and REINFORCE (RE). *: significantly different from SL with $p < 0.001$

5 Error Propagation Experiment

We hypothesized that reinforcement learning avoids error propagation. In this section, we describe our algorithm and the experiment that identifies error propagation in the arc-standard parsers.

5.1 Error Propagation

Section 3.1 explained that a transition-based parser goes through the sentence incrementally and must select a transition from [SHIFT, LEFT_l,

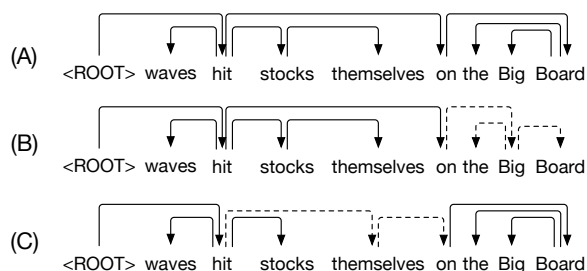


Figure 2: Three dependency graphs: gold (A), arc errors caused by one decision error (B) and arc errors caused by multiple decision errors (C).

RIGHT_l] at each step. We use the term *arc error* to refer to an erroneous arc in the resulting tree. The term *decision error* refers to a transition that leads to a loss in parsing accuracy. Decision errors in the parsing process lead to one or more arc errors in the resulting tree. There are two ways in which a single decision error may lead to multiple arc errors. First, the decision can deterministically lead to more than one arc error, because (e.g.) an erroneously formed arc also blocks other correct arcs. Second, an erroneous parse decision changes some of the features that the model uses for future decisions and these changes can lead to further (decision) errors down the road.

We illustrate both cases using two incorrect derivations presented in Figure 2. The original gold tree is repeated in (A). The dependency graph in Figure 2 (B) contains three erroneous dependency arcs (indicated by dashed arrows). The first error must have occurred when the parser executed RIGHT_{amod} creating the arc *Big* → *Board*. After this error, it is impossible to create the correct relations *on* → *Board* and *Board* → *the*. The wrong arcs *Big* → *the* and *on* → *Big* are thus all the result of a single decision error.

Figure 2 (C) represents the dependency graph that is actually produced by our parser.⁶ It contains two erroneous arcs: *hit* → *themselves* and *themselves* → *on*. Table 4 provides a possible sequence of steps that led to this derivation, starting from the moment *stocks* was added to the stack (Step 4). The first error is introduced in Step 5', where *hit* combines with *stocks* before *stocks* has picked up its dependent *themselves*. At that point, *themselves* can no longer be combined with the right head. The proposition *on*, on the other hand, can

⁶The example is a fragment of a more complex sentence consisting of 33 tokens. The parser does provide the correct output when it analyzes this sequence in isolation.

Step	Transition	Stack	Buffer	Arcs
4	SHIFT	<ROOT> <i>hit stocks</i>	<i>themselves on the Big Board</i>	A_1
5'	RIGHT _{dobj}	<ROOT> <i>hit</i>	<i>themselves on the Big Board</i>	$A_2 = A_1 \cup$ $\{hit \xrightarrow{dobj} stocks\}$
6'	SHIFT	<ROOT> <i>hit themselves</i>	<i>on the Big Board</i>	A_2
7'	SHIFT	<ROOT> <i>hit themselves on</i>	<i>the Big Board</i>	A_2
...				
10'	SHIFT	<ROOT> <i>hit themselves on the Big Board</i>		A_2
11'	LEFT _{nn}	<ROOT> <i>hit themselves on the Board</i>		$A_3 = A_2 \cup$ $\{Board \xrightarrow{nn} Big\}$
12'	LEFT _{det}	<ROOT> <i>hit themselves on Board</i>		$A_4 = A_3 \cup$ $\{Board \xrightarrow{det} the\}$
13'	RIGHT _{pobj}	<ROOT> <i>hit themselves on</i>		$A_5 = A_4 \cup$ $\{on \xrightarrow{pobj} Board\}$
14'	RIGHT _{dep}	<ROOT> <i>hit themselves</i>		$A_6 = A_5 \cup$ $\{themselves \xrightarrow{dep} on\}$

Table 4: Possible parsing walk-through with error

still be combined with the correct head. This error is introduced in Step 7', where the parser moves *on* to the stack rather than creating an arc from *hit* to *themselves*.⁷ There are thus two decision errors that lead to the arc errors in Figure 2 (C). The second decision error can, however, be caused indirectly by the first error. If a decision error causes additional decision errors later in the parsing process, we talk of error propagation. This cannot be known just by looking at the derivation.

5.2 Examining the impact of decision errors

We examine the impact of individual decision errors on the overall parse results in our test set by combining a dynamic oracle and a recursive function. We use a dynamic oracle based on Goldberg et al. (2014) which gives us the overall loss at any point during the derivation. The loss is equal to the minimal number of arc errors that will have been made once the parse is complete. We can thus deduce how many arc errors are deterministically caused by a given decision error.

The propagation of decision errors cannot be determined by simply examining the increase in loss during the parsing process. We use a recursive function to identify whether a particular parse suffered from this. While parsing the sentence, we register which decisions lead to an increase in loss. We then recursively reparse the sentence correcting one additional decision error during each run until the parser produces the gold. If each erroneous decision has to be corrected in order to arrive at the gold, we assume the decision errors are

⁷Note that technically, *on* can still become a dependent of *hit*, but this can only happen if *on* becomes the head of *themselves* which would also be an error.

	SL	RL-O	RL-R	RL-M
Total Loss	7069	6227	6042	6144
Dec. Errors	5177	4410	4345	4476
Err. Prop.	1399	1124	992	1035
New errors	411	432	403	400
Loss/error	1.37	1.41	1.39	1.37
Err. Prop. (%)	27.0	25.5	22.8	23.1

Table 5: Overview of average impact of decision errors

independent of each other. If, on the other hand, the correction of a specific decision also fixes other decisions down the road, the original parse suffers from error propagation.

The results are presented in Table 5. *Total Loss* indicates the number of arc errors in the corpus, *Dec. Errors* the number of decision errors and *Err. Prop.* the number of decision errors that were the result of error propagation. This number was obtained by comparing the number of decision errors in the original parse to the number of decision errors that needed to be fixed to obtain the gold parse. If less errors had to be fixed than originally present, we counted the difference as error propagation. Note that fixing errors sometimes leads to new decision errors during the derivation. We also counted the cases where more decision errors needed to be fixed than were originally present and report them in Table 5.⁸

⁸We ran an alternative analysis where we counted all cases where fixing one decision error in the derivation reduced the overall number of decision errors in the parse by more than one. Under this alternative analysis, similar reductions in the proportion of error propagation were observed for reinforcement learning.

On average, decision errors deterministically lead to more than one arc error in the resulting parse tree. This remains stable across systems (around 1.4 arc errors per decision error). We furthermore observe that the proportion of decision errors that are the result of error propagation has indeed reduced for all reinforcement learning models. Among the errors avoided by APG, 35.9% were propagated errors for RL-ORACLE, 48.9% for RL-RANDOM, and 51.9% for RL-MEMORY. These percentages are all higher than the proportion of propagated errors occurring in the corpus parsed by SL (27%). This outcome confirms our hypothesis that reinforcement learning is indeed more robust for making decisions in imperfect environments and therefore reduces error propagation.

6 Conclusion

This paper introduced Approximate Policy Gradient (APG), an efficient reinforcement learning algorithm for NLP, and applied it to a high-performance greedy dependency parser. We hypothesized that reinforcement learning would be more robust against error propagation and would hence improve parsing accuracy.

To verify our hypothesis, we ran experiments applying APG to three transition systems and two languages. We furthermore introduced an experiment to investigate which portion of errors were the result of error propagation and compared the output of standard supervised machine learning to reinforcement learning. Our results showed that: (a) reinforcement learning indeed improved parsing accuracy and (b) propagated errors were over-represented in the set of avoided errors, confirming our hypothesis.

To our knowledge, this paper is the first to show experimentally that reinforcement learning can reduce error propagation in an NLP task. This result was obtained by a straight-forward implementation of reinforcement learning. Furthermore, we only applied reinforcement learning in the training phase, leaving the original efficiency of the model intact. Overall, we see the outcome of our experiments as an important first step in exploring the possibilities of reinforcement learning for tackling error propagation.

Recent research on parsing has seen impressive improvement during the last year achieving UAS around 94% (Andor et al., 2016). This improve-

ment is partially due to other approaches that, at least in theory, address error propagation, such as beam search. Both the reinforcement learning algorithm and the error propagation study we developed can be applied to other parsing approaches. There are two (related) main questions to be addressed in future work in the domain of parsing. The first addresses whether our method is complementary to alternative approaches and could also improve the current state-of-the-art. The second question would address the impact of various approaches on error propagation and the kind of errors they manage to avoid (following Ng and Curran (2015)).

APG is general enough for other structured prediction problems. We therefore plan to investigate whether we can apply our approach to other NLP tasks such as coreference resolution or semantic role labeling and investigate if it can also reduce error propagation for these tasks.

The source code of all experiments is publicly available at <https://bitbucket.org/cttl1/redep-java>.

Acknowledgments

The research for this paper was supported by the Netherlands Organisation for Scientific Research (NWO) via the Spinoza-prize Vossen projects (SPI 30-673, 2014-2019) and the VENI project *Reading between the lines* (VENI 275-89-029). Experiments were carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. We would like to thank our friends and colleagues Piek Vossen, Roser Morante, Tommaso Caselli, Emiel van Miltenburg, and Ngoc Do for many useful comments and discussions. We would like to extend our thanks the anonymous reviewers for their feedback which helped improving this paper. All remaining errors are our own.

References

- Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-Based Parsing and Tagging with Neural Networks. In *EMNLP 2015*, pages 1354–1359. ACL.
- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. *arXiv.org*, cs.CL.

- Jonathan Berant and Percy Liang. 2015. Imitation Learning of Agenda-based Semantic Parsers. *TACL*, 3:545–558.
- Anders Björkelund and Joakim Nivre. 2015. Non-Deterministic Oracles for Unrestricted Non-Projective Transition-Based Dependency Parsing. In *IWPT 2015*, pages 76–86. ACL.
- Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *ICML 2015*.
- Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *EMNLP 2014*, pages 740–750. ACL.
- Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based Structured Prediction. *Machine Learning*, 75(3):297–325, 6.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *ACL 2015*, pages 334–343.
- Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. In *ACL 2002*, pages 239–246. ACL.
- Yoav Goldberg and Joakim Nivre. 2012. A Dynamic Oracle for Arc-Eager Dependency Parsing. In *COLING 2012*, pages 959–976.
- Yoav Goldberg and Joakim Nivre. 2013. Training Deterministic Parsers with Non-Deterministic Oracles. In *TACL 2013*, volume 1, pages 403–414.
- Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. In *TACL 2014*, volume 2, pages 119–130.
- Carlos Gomez-Rodriguez, Francesco Sartorio, and Giorgio Satta. 2014. A Polynomial-Time Dynamic Oracle for Non-Projective Dependency Parsing. In *EMNLP 2014*, pages 917–927. ACL.
- Alvin C. Grissom II, Jordan Boyd-Graber, He He, John Morgan, and Hal Daume III. 2014. Don’t Until the Final Verb Wait: Reinforcement Learning for Simultaneous Machine Translation. In *EMNLP 2014*, pages 1342–1352.
- Dan Han, Pascual Martínez-Gómez, Yusuke Miyao, Katsuhito Sudoh, and Masaaki Nagata. 2013. Effects of parsing errors on pre-reordering performance for Chinese-to-Japanese SMT. *PACLIC 27*, pages 267–276.
- Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *EMNLP 2015*, pages 1373–1378. ACL.
- Jiarong Jiang, Adam Teichert, Hal Daumé III, and Jason Eisner. 2012. Learned Prioritization for Trading Off Accuracy and Speed. *ICML workshop on Inference: Interactions between Inference and Learning*, (0964681):1–9.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *CoRR*, abs/1603.0.
- Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware Ensemble Training for Semi-supervised Dependency Parsing. In *ACL 2014*, pages 457–467.
- Francis Maes, Ludovic Denoyer, and Patrick Gallinari. 2009. Structured prediction with reinforcement learning. *Machine Learning*, (77):271–301.
- Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In *EMNLP-CoNLL 2007*.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP 2005*, pages 523–530. Association for Computational Linguistics.
- Dominick Ng and James R Curran. 2015. Identifying Cascading Errors using Constraints in Dependency Parsing. In *ACL-IJCNLP*, pages 1148–1158, Beijing. ACL.
- Joakim Nivre and Daniel Fernández-González. 2014. Arc-eager Parsing with the Tree Constraint. *Computational Linguistics*, 40(2):259–267, 6.
- Joakim Nivre, Johan Hall, and Jens Nilsson. 2006a. MaltParser: A data-driven parser-generator for dependency parsing. In *LREC 2006*, volume 6, pages 2216–2219.
- Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *CoNLL 2006*, pages 221–225. ACL.
- Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Eryiğit Gülşen, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.
- Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *IWPT 2003*, pages 149–160.
- Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

- Joakim Nivre. 2009. Non-projective Dependency Parsing in Expected Linear Time. In *ACL-IJCNLP 2009*, pages 351–359, Stroudsburg, PA, USA. ACL.
- Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *EMNLP 2006*, pages 62–69, Sydney, Australia. ACL.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. *ICLR*, pages 1–15.
- Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *AISTATS*, 15:627–635.
- Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-Rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109.
- Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum Risk Training for Neural Machine Translation. In *ACL 2016*, pages 1683–1692, Berlin, Germany. ACL.
- Hyun-Je Song, Jeong-Woo Son, Tae-Gil Noh, Seong-Bae Park, and Sang-Jo Lee. 2012. A Cost Sensitive Part-of-Speech Tagging: Differentiating Serious Errors from Minor Errors. In *ACL 2012*, pages 1025–1034. ACL.
- Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS 1999*, pages 1057–1063.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured Training for Neural Network Transition-Based Parsing. In *ACL-IJCNLP 2015*, pages 323–333. ACL.
- Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.
- Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of IWPT*, pages 195–206.
- Bishan Yang and Claire Cardie. 2013. Joint Inference for Fine-grained Opinion Extraction. In *ACL 2013*, pages 1640–1649. ACL.
- Lidan Zhang and Kwok Ping Chan. 2009. Dependency Parsing with Energy-based Reinforcement Learning. In *IWPT 2009*, pages 234–237. ACL.

Noisy-context surprisal as a human sentence processing cost model

Richard Futrell and Roger Levy
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
{futrell, rplevy}@mit.edu

Abstract

We use the noisy-channel theory of human sentence comprehension to develop an incremental processing cost model that unifies and extends key features of expectation-based and memory-based models. In this model, which we call **noisy-context surprisal**, the processing cost of a word is the surprisal of the word given a noisy representation of the preceding context. We show that this model accounts for an outstanding puzzle in sentence comprehension, language-dependent structural forgetting effects (Gibson and Thomas, 1999; Vasishth et al., 2010; Frank et al., 2016), which are previously not well modeled by either expectation-based or memory-based approaches. Additionally, we show that this model derives and generalizes locality effects (Gibson, 1998; Demberg and Keller, 2008), a signature prediction of memory-based models. We give corpus-based evidence for a key assumption in this derivation.

1 Introduction

Models of human sentence processing difficulty can be divided into two kinds, **expectation-based** and **memory-based**. Expectation-based models predict the processing difficulty of a word from the word’s surprisal given previous material in the sentence (Hale, 2001; Levy, 2008a). These models have good coverage: they can account for effects of syntactic construction frequency and resolution of ambiguity on incremental processing difficulty. Memory-based models, on the other hand, explain difficulty resulting from working memory limitations during incremental parsing (Gibson, 1998;

Lewis and Vasishth, 2005); a major prediction of these models is **locality effects**, where processing a word is difficult when it is far from other words with which it must be syntactically integrated. Expectation-based models do not intrinsically capture this difficulty.

Integrating these two approaches at a high level has proven challenging. A major hurdle is that the theories are typically stated at different levels of analysis: expectation-based theories are computational-level theories (Marr, 1982) specifying what computational problem the human sentence processing system is solving—the problem of how update one’s belief about a sentence given a new word—without specifying implementation details. Memory-based theories such as Lewis and Vasishth (2005) are for the most part based in mechanistic algorithmic-level theories describing the actions of a specific incremental parser.

Previous theories that capture both surprisal and locality effects have typically done so by augmenting parsing models with a special prediction-verification operation to capture surprisal effects (Demberg and Keller, 2009; Demberg et al., 2013), or by combining surprisal and memory-based cost derived from a parsing model as separate factors in a linear model (Shain et al., 2016). These models capture surprisal and locality effects at the same time, but they do not clearly capture phenomena involving the interaction of memory and probabilistic expectations such as language-dependent structural forgetting (see Section 3).

Here we develop a computational-level model capturing both memory and expectation effects from a single set of principles, without reference to a specific parsing algorithm. In our model, the processing cost of a word is a function of its surprisal given a *noisy* representation of previous context (Section 2). We show that the model can reproduce structural forgetting effects, including

the difference between English and German (Section 3), a phenomenon not previously captured by memory-based or expectation-based models in isolation. We also give a derivation of the existence of locality effects in the model; these effects were previously accounted for only in mechanistic memory-based models (Section 4). The derivation yields a generalization of classic locality effects which we call **information locality**: sentences are predicted to be easier to process when words with high mutual information are close. We give corpus-based evidence that words in syntactic dependencies have high mutual information, meaning that classical dependency locality effects can be seen as a subset of information locality effects.

2 Noisy-Context Surprisal

In surprisal theory, the processing cost of a word is asserted to be proportional to extent to which one must change one’s beliefs given that word (Hale, 2001; Smith and Levy, 2013). So the cost of a word is (up to proportionality):

$$C_{\text{surprisal}}(w_i|w_{1:i-1}) = -\log p_L(w_i|w_{1:i-1}), \quad (1)$$

where $p_L(\cdot|\cdot)$ is the conditional probability of a word in context in a probabilistic language L .

Standard surprisal assumes that the comprehender has perfect access to a representation of w_i ’s full context, including the words preceding it in the sentence ($w_{1:i-1}$) and also extra-sentential context (which we leave implicit). But given that human working memory is limited, the assumption of perfect access is unrealistic. We propose that processing cost at a word is better modeled as the cost of belief updates given a *noisy representation* of the previous input. The probability of a word given a noisy context is modeled as the noisy channel probability of the word, assuming that people do noisy channel inference on their context representation (Levy, 2008b; Gibson et al., 2013). Given this model, the expected processing cost of a word is its expected surprisal over the possible noisy representations of its context.

The noisy-context surprisal processing cost function is thus:¹

$$C(w_i|w_{1:i-1}) = \mathbb{E}_{V|w_{1:i-1}} [-\log p_L^{\text{NC}}(w_i|V)] \quad (2)$$

$$= -\sum_V p_N(V|w_{1:i-1}) \log p_L^{\text{NC}}(w_i|V) \quad (3)$$

¹Neglecting the implicit proportionality term in Equation 1.

where V is the noisy representation of the previous material $w_{1:i-1}$, the **noise distribution** p_N characterizes how memory of previous material may be corrupted, and $p_L^{\text{NC}}(\cdot|\cdot)$ is the noisy-channel probability of a word given a noisy context, computed via marginalization:

$$p_L^{\text{NC}}(w_i|V) = \sum_{w_{1:i-1}} p_L(w_i|w_{1:i-1}) p^{\text{NC}}(w_{1:i-1}|V)$$

with $p^{\text{NC}}(w_{1:i-1}|V)$ computed via Bayes Rule:

$$p^{\text{NC}}(w_{1:i-1}|V) \propto p_N(V|w_{1:i-1}) p_L(w_{1:i-1}).$$

Note here that w_i ’s cost is computed using its true identity but a noisy representation of the context: from the incremental perspective, w_i is observed now, but context is stored and retrieved in a potentially noisy storage medium. This asymmetry between noise levels for proximal versus distal input differs from the noisy-channel surprisal model of Levy (2011), and is crucial to the derivation of information locality we present in Section 4.

Here we use two types of noise distributions for p_N : erasure noise and deletion noise. In **erasure noise**, a symbol in the context is probabilistically erased and replaced with a special symbol \mathbb{E} with probability e . In **deletion noise**, a symbol is erased from the sequence completely, leaving no trace. Given deletion noise, a comprehender does not know how many symbols were in the original context; with erasure noise, the comprehender knows exactly which symbols were affected by noise. In both cases, we assume that the application or non-application of noise is probabilistically independent among elements in the context. We use these concrete noise distributions for convenience, but we believe our results should generalize to larger classes of noise distributions.

3 Structural Forgetting Effects

Here we show that noisy-context surprisal as a processing cost model can reproduce effects that were not previously well-explained by either expectation-based or memory-based theories. In particular, we take up the puzzle of **structural forgetting effects**, where comprehenders seem to forget structural elements of a sentence prefix when predicting the rest of the sentence. The result is that some ungrammatical sentences have lower processing cost and higher acceptability than some complex grammatical sentences: with

doubly nested relative clauses, for instance, subjects rate ungrammatical sentence (1) as more acceptable than sentence (2), forgetting about the VP predicted by the second noun (Gibson and Thomas, 1999).

(1) *The apartment₁ that the maid₂ who the cleaning service₃ had₃ sent over was₁ well-decorated.

(2) The apartment₁ that the maid₂ who the cleaning service₃ had₃ sent over was₂ cleaning every week was₁ well-decorated.

Vasishth et al. (2010) show this same effect in reading times at the last verb: in English native speakers are more surprised to encounter a third VP than not to. However, this effect is language-specific: the same authors find that in German, native speakers are more surprised when a third VP is missing than when it is present. Frank et al. (2016) show further that native speakers do not show the effect in Dutch, but Dutch-native L2 speakers of English do show the effect in English. The result shows that the memory resources taxed by these structures are themselves meaningfully shaped by the distributional statistics of the language.

The verb forgetting effect is a challenge for both expectation-based and memory-based models. Pure expectation-based models cannot reproduce the effect: they have no mechanism for forgetting an established VP prediction and thus they assign small or zero probability to ungrammatical sentences. On the other hand, memory-based models will have to account for why the same structures are forgotten in English but not in German. Here we show that noisy-context surprisal provides the first purely computational-level account for the language-dependent verb forgetting effect. The essential mechanism is that when verb-final nested structures are more probable in a language, then they will be better preserved in a noisy memory representation.

3.1 Model of Verb Forgetting

Table 1 presents a toy probabilistic context-free grammar for the constructions involved in verb forgetting. The grammar generates strings over the alphabet of N (noun), V (verb), C (complementizer), P (preposition). We apply deletion noise with by-symbol deletion probability d . So for example, given a prefix NCNCNVV, the prefix can be corrupted to NCNNVV with probability proportional to d , representing one deletion. In that

Rule	Probability
$S \rightarrow NP V$	1
$NP \rightarrow N$	$1 - m$
$NP \rightarrow N RC$	mr
$NP \rightarrow N PP$	$m(1 - r)$
$PP \rightarrow P NP$	1
$RC \rightarrow C V NP$	s
$RC \rightarrow C NP V$	$1 - s$

Table 1: Toy grammar used to demonstrate verb forgetting. Nouns are postmodified with probability m ; a postmodifier is a relative clause with probability r , and a relative clause is V-initial with probability s . For practical reasons we bound non-terminal rewrites of NP at 2.

case a noisy-channel comprehender might incorrectly infer that the original prefix was in fact NCNPNVV, and thus fail to predict a third verb.

To illustrate that noisy surprisal can account for language-dependent verb forgetting, we show in Figure 1 the differences between noisy surprisal values for grammatical (V) and ungrammatical (end-of-sentence) continuations of prefixes NCNCNVV under parameter settings reflecting the difference between English and German, and compare these differences with self-paced reading times observed after the final verb by Vasishth et al. (2010). Noisy surprisal qualitatively reproduces language-dependent verb forgetting: in English the ungrammatical continuation is higher surprisal, but in German the grammatical continuation is higher surprisal. The English–German difference in the model is entirely accounted for by the parameter s , which determines the proportion of relative clauses that are verb-initial. In English, most relative clauses are subject-extracted and those are verb-initial, so for English $s \approx .8$ (Roland et al., 2007). German, in contrast, has $s \approx 0$, since its relative clauses are obligatorily verb-final. When verb-final relative clauses have higher prior probability, a doubly-nested RC prefix NCNCVV is more likely to be preserved by a rational noisy-channel comprehender.

The results of Figure 1 do not speak, however, to the generality of the model’s predictions regarding verb forgetting. To explore this matter, we partition the model’s four-dimensional parameter space into regions distinguishing whether noisy-context surprisal is lower for (G) grammatical continuations or (U) ungrammatical contin-

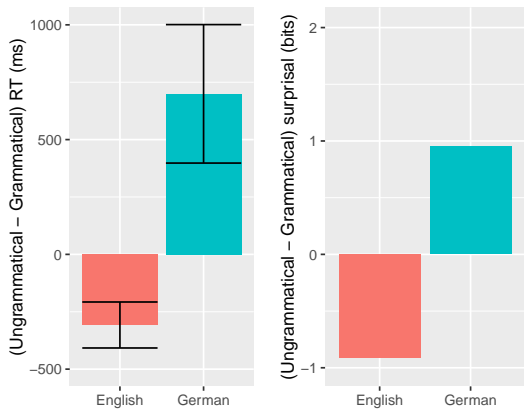


Figure 1: Differences in reaction times for ungrammatical continuations minus grammatical continuations, compared to noisy surprisal differences. RT data comes from self-paced reading experiments in Vasishth et al. (2010) in the post-VP region. The noisy surprisal predictions are produced with $d = .2$, $m = .5$, $r = .5$ fixed, and $s = .8$ for English and $s = 0$ for German.

uations for (1) singly-embedded NCNV and (2) doubly-embedded NCNCNVV contexts. Figure 2 shows this partition for a range of r , s , m , and d . In the blue region, grammatical continuations are lower-cost than ungrammatical continuations for both singly and doubly embedded contexts, as in German (G_1G_2); in the red region, the ungrammatical continuation is lower-cost for both contexts (U_1U_2). In the green region, the grammatical continuation is lower cost for single embedding, but higher cost for double embedding, as in English (G_1U_2). No combination of parameter values instantiates U_1G_2 (for either the depicted or other possible values of m and d). Thus both the English and German behavioral patterns are quite generally predicted by the model. Furthermore, each language’s statistics place it in a region of parameter space plausibly corresponding to its behavioral pattern: the English-type forgetting effect is predicted mostly for high s , the German-type for low s .

The only previous formalized account of language-specific verb forgetting, Frank et al. (2016), showed that Simple Recurrent Networks (SRNs) trained on English and Dutch data partly reproduce the verb forgetting effect in the surprisals they assign to the final verb. Our model provides an explanation of the SRN’s behavior. When an SRN predicts words, it effectively uses

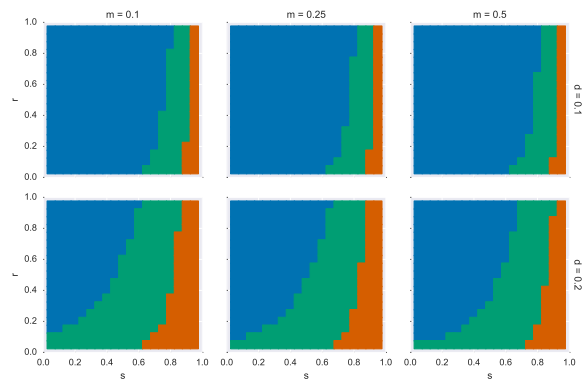


Figure 2: Regions of different model behavior with respect to parameters r , s , m , and d (see Table 1). Blue: G_1G_2 ; red: U_1U_2 ; green: G_1U_2 (see text).

a lossily compressed representation of the previous words. This lossy compression is analogous to the noisy representation posited here.

4 Information Locality

Here we show how, given an appropriate noise distribution, noisy surprisal gives rise to locality effects. Standard locality effects are related to syntactic dependencies: the claim is that processing is difficult when the parser must make a syntactic connection with an element that has been in memory for a long time. In Section 4.1, we derive a more general prediction: that processing is difficult when any elements with high mutual information are far from one another. The effect arises under noisy surprisal because context elements that would have been helpful for predicting a word might have been forgotten. We call this principle **information locality**. In Section 4.3, we argue that words in syntactic dependencies have higher mutual information than other word pairs, which leads to a view of dependency locality effects as a special case of information locality effects.

4.1 Derivation of Information Locality

Viewing processing cost as a function of word order, noisy surprisal gives rise to the generalization that cost is minimized when elements with high mutual information are close. We show this by decomposing the noisy surprisal cost of a word into many terms of higher-order mutual information with the context, then showing that applying a certain kind of erasure noise to the context causes

these terms to be downweighted based on their distance to the word. Thus the best word order puts the words that have high mutual information with a word close to that word.

4.1.1 Noise Distribution

Noisy surprisal gives rise to information locality under a family of noise distributions which we call **progressive erasure noise**, which is any noise function that erases discrete elements of a sequence with increasing probability the earlier those elements are in the sequence. Formally, in progressive erasure noise, the i th element in a sequence X with length $|X|$ is erased with probability proportional to some monotonically increasing function of how far left that element is in the sequence: $f(|X| - i)$. As a concrete example of progressive erasure noise, consider an exponential decay function, such that the probability that an element i in X remains unerased is $(1 - e)^{|X| - i}$ for some probability e . The exponential decay function corresponds to a noise model where the context sequence is hit with erasure noise successively as each word is processed. Any progressive erasure noise distribution suffices for the derivation here to go through.

4.1.2 Decomposing Surprisal Cost

In noisy surprisal theory, the cost of a word w_i in context $w_{1:i-1}$ is:

$$\begin{aligned} C(w_i|w_{1:i-1}) &= \mathbb{E}_{V|w_{1:i-1}} [-\log p(w_i|V)] \\ &= \mathbb{E}_{V|w_{1:i-1}} [h(w_i) - \text{pmi}(w_i; V)] \\ &= h(w_i) - \mathbb{E}_{V|w_{1:i-1}} [\text{pmi}(w_i; V)], \end{aligned} \quad (4)$$

where $h(\cdot)$ is surprisal (here unconditional, equivalent to log inverse-frequency) and $\text{pmi}(\cdot; \cdot)$ is **pointwise mutual information** between two values under a joint distribution:

$$\text{pmi}(x; y) = h(x) + h(y) - h(x, y). \quad (5)$$

Essentially, each word has an inherent cost determined by its log inverse probability, mitigated to the extent that it is predictable from context ($\text{pmi}(w_i; w_{1:i-1})$).

Now define the **interaction information** between a sequence of m values $\{a\}$ drawn from a sequence of m random variables $\{\alpha\}$ (McGill,

1955; Bell, 2003) as:²

$$i(a_1; \dots; a_m) = \sum_{n=1}^m \sum_{I \in \binom{1:m}{n}} (-1)^{m-n-1} h(a_{I_1}, \dots, a_{I_n}),$$

where the notation $\binom{1:m}{n}$ means all cardinality- n subsets of the set of integers 1 through m . The equation amounts to alternately adding and subtracting the joint surprisals of all subsets of values. For $m = 2$, expanding the equation reveals that mutual information is a special case of interaction information.

Supposing that the noisy representation of context V is the result of running the veridical context $w_{1:i-1}$ through progressive erasure noise, we can see V as a sequence of values $v_{1:i-1}$, where each v_i is equal to either w_i or the erasure symbol E . Rewriting $\text{pmi}(w_i; V)$ as $\text{pmi}(w_i; v_{1:i-1})$, we can decompose it into interaction informations as follows:

$$\text{pmi}(w_i; v_{1:i-1}) = \sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(w_i; v_{I_1}; \dots; v_{I_n}). \quad (6)$$

The equation expresses a sum of interaction informations between the current word w_i and all subsets of the context values.³

²Higher-order information terms are typically defined using a different sign convention and referred to as **coinformation** or **multivariate mutual information** (Bell, 2003). For even orders, interaction information is equal to coinformation. For odd orders, interaction information is equal to negative coinformation. We adopt our particular sign convention to make the generalization of information locality easier to express.

³To see that this is true, first note that we can express joint surprisal in terms of interaction information:

$$h(a_1, \dots, a_m) = - \sum_{n=1}^m \sum_{I \in \binom{1:m}{n}} i(a_{I_1}; \dots; a_{I_n}).$$

Now consider the pmi of a value a_i with a sequence $a_{1:i-1}$. Using the decomposition of joint surprisal to expand the definition of pmi in Equation 5, we get:

$$\begin{aligned} \text{pmi}(a_i; a_{1:i-1}) &= h(a_i) + h(a_{1:i-1}) - h(a_i, a_{1:i-1}) \\ &= h(a_i) + h(a_{1:i-1}) - h(a_{1:i}) \\ &= h(a_i) - \sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(a_{I_1}; \dots; a_{I_n}) \\ &\quad + \sum_{n=1}^i \sum_{I \in \binom{1:i}{n}} i(a_{I_1}; \dots; a_{I_n}) \end{aligned}$$

In the final expression, all the terms that do not contain a_i

Now combining Equations 4 and 6, we get:

$$\begin{aligned}
C(w_i|w_{1:i-1}) &= h(w_i) - \\
&\mathbb{E}_{v|w_{1:i-1}} \left[\sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(w_i; v_{I_1}; \dots; v_{I_n}) \right] \\
&= h(w_i) - \sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} \sum_v p_N(v|w_{1:i-1}) i(w_i; v_{I_1}; \dots; v_{I_n}).
\end{aligned}$$

Now if any element of an interaction information term is \mathbb{E} , then that whole interaction information term is equal to 0. This happens because the probability that an element is erased is independent of the identity of other elements in the sequence, and thus \mathbb{E} has no interaction information with any subset of those elements. That is, $i(w_i; v_{I_1}; \dots; v_{I_n}) = 0$ unless $v_{I_j} = w_{I_j}$ for all j . This allows us to write:

$$\begin{aligned}
C(w_i|w_{1:i-1}) &= h(w_i) - \\
&\sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(w_i; w_{I_1}; \dots; w_{I_n}) \sum_{m \in \{0,1\}^{i-1}} p_N(m) m_I
\end{aligned}$$

where the variable m ranges over bit-masks of length $i-1$, and m_I is equal to 1 when all indices I in m are equal to 1, and 0 otherwise. Now $\sum_{m \in \{0,1\}^{i-1}} p_N(m) m_I$ is the total probability that all of a set of indices I survives erasure. Thus, informally:

$$\begin{aligned}
C(w_i|w_{1:i-1}) &= h(w_i) - \\
&\sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} p_N(I \text{ survives}) i(w_i; w_{I_1}; \dots; w_{I_n}).
\end{aligned} \tag{7}$$

That is, the cost of a word is its inherent cost minus its interaction informations with context, which are weighted by the probability that all elements of those interactions survive erasure.

cancel out, leaving:

$$\begin{aligned}
\text{pmi}(a_i; a_{1:i-1}) &= h(a_i) + \sum_{n=0}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(a_i; a_{I_1}; \dots; a_{I_n}) \\
&= h(a_i) + \sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(a_i; a_{I_1}; \dots; a_{I_n}) - h(a_i) \\
&= \sum_{n=1}^{i-1} \sum_{I \in \binom{1:i-1}{n}} i(a_i; a_{I_1}; \dots; a_{I_n}),
\end{aligned}$$

which gives Equation 6 when applied to w_i and $v_{1:i-1}$.

Under progressive erasure noise, the probability that a subset of variables is erased increases the farther left those variables are in the context. Therefore, Equation 7 expresses information locality: context elements which are predictive of w_i will only get to mitigate the cost of processing w_i if they are close to it. The surprisal-mitigating effect of a context element on a word w_i decreases as that element gets farther from w_i .

4.2 Noisy Surprisal and Dependency Locality

Memory-based models of sentence processing account for apparent **dependency locality effects**, which is processing cost apparently arising from two words linked in a syntactic dependency appearing far from one another (Gibson, 1998). Dependency length has been proposed as a rough measure of comprehension and production difficulty, and studied as a predictor of reaction times (Grodner and Gibson, 2005; Demberg and Keller, 2008; Mitchell et al., 2010; Shain et al., 2016), and also as a theory of production preferences and linguistic typology, under the assumption that people prefer to produce sentences with short dependencies (dependency length minimization) (Hawkins, 1994; Gildea and Temperley, 2010; Futrell et al., 2015; Rajkumar et al., 2016).

Dependency locality follows from information locality if words linked in a syntactic dependency have particularly high mutual information. To see this, consider only the lowest-order interaction information terms in Equation 7, truncating the summation over n at 1. We can write

$$C(w_i|w_{1:i-1}) = h(w_i) - \sum_{j=1}^{i-1} f(i-j) \text{pmi}(w_i; w_j) + R,$$

where R collects all the interaction information terms of order greater than 2, and $f(d)$ is the monotonically decreasing survival probability of a d -back word, described in Section 4.1.1. The effects of R are bounded because higher-order mutual information terms are more penalized by erasure noise than lower-order terms, simply because large sets of context items are more likely to experience at least one erasure.

If the effects of R are negligible, then the cost of a whole utterance w as a function of word order is determined only by pairwise information locality:

$$C(w) \approx \sum_{i=1}^{|w|} h(w_i) - \sum_{i=2}^{|w|} \sum_{j=1}^{i-1} f(i-j) \text{pmi}(w_i; w_j).$$

If words linked in a dependency have higher mutual information than words that are not, then the processing cost as a function of word order is a monotonically increasing function of dependency length. Under this assumption, for which we provide evidence below, dependency locality effects can be seen as a special case of information locality effects. As a theory of production preferences or typology, processing cost as a monotonically increasing function of dependency length suffices to derive the predictions of dependency length minimization (Ferrer i Cancho, 2015).

4.3 Mutual Information and Syntactic Dependency

We have shown that noisy-context surprisal derives information locality, and argued that dependency locality can be seen as a special case of information locality. However, deriving dependency locality requires a crucial assumption that words linked in a dependency have higher mutual information than those words that are not.

To test this assumption, we calculated mutual information between wordforms in various dependency relations in the Google Syntactic n -gram corpus (Goldberg and Orwant, 2013). We compared the mutual information of content words in a direct dependency relationship to content words in grandparent–grandchild and sister–sister dependency relationships. Mutual information was estimated using maximum likelihood estimation from frequencies, treating the corpus as samples from a distribution over (head, dependent) pairs. In order to exclude nonlinguistic forms, we only included wordforms if they were among the top 10000 most frequent wordforms in the corpus. The direct head–dependent frequencies were calculated from the same corpus as the grandparent-grandchild frequencies, so that all mutual information estimates are affected by the same frequency cutoff. The results are shown in Table 2: direct head–dependent pairs indeed have the highest mutual information.

To test the crosslinguistic validity of this generalization about syntactic dependency and mutual information, we calculated mutual information between the distributions over POS tags for dependency pairs of 43 languages in the Universal Dependencies corpus (Nivre et al., 2016). For this calculation, we used mutual information over POS tags rather than wordforms to avoid data sparsity issues. The results are shown in Figure 3.

Relation	MI (bits)
Head–dependent	1.79
Grandparent–dependent	1.34
Sister–sister	1.19

Table 2: Mutual information over wordforms in different dependency relations in the Syntactic n -gram corpus. The pairwise comparison of head–dependent and grandparent–dependent MI is significant at $p < 0.005$ by Monte Carlo permutation tests over n -grams with 500 samples. The comparison of head–dependent and sister–sister MI is not significant.

Again, we find that mutual information is highest for direct head–dependency pairs, and falls off for more distant relations. These results show that two words in a syntactic dependency relationship are more predictive of each other than two words in some other kinds of relationship.

We also compared the mutual information of word pairs in and out of dependency relationships while controlling for distance. This test has a dual purpose. First, it allows us to control for distance when claiming that words in dependency relationships have high mutual information. Second, it allows us to test a simple prediction of information locality as applied to language production: that words with high mutual information should be close together. For pairs of words (w_i, w_{i+k}) , we calculated the pmi values among POS tags of the words. Figure 4 shows the average pmi of all words at each distance compared with the average pmi of the subset of words in a direct dependency relationship at that distance. In all languages, we find that words in a dependency relationship have higher pmi than the baseline, especially at close distances. Furthermore, we find that words at close distances tend to have higher pmi, regardless of whether they are in a dependency relationship.

4.4 Discussion

Information locality can be seen as a decay in the effectiveness of contextual cues for predicting words. Precisely such a decay in cue effectiveness was found to be effective for predicting entropy distributions across sentences in Qian and Jaeger (2012), although that work did not distinguish between an inherent, noise-based decay in cue effectiveness or optimized placement of cues.



Figure 3: Mutual information over POS tags for dependency relations in the Universal Dependencies 1.4 corpus, for languages with over 500 sentences. All pairwise MI comparisons are significant at $p < 0.005$ by Monte Carlo permutation tests over dependency observations with 500 samples.

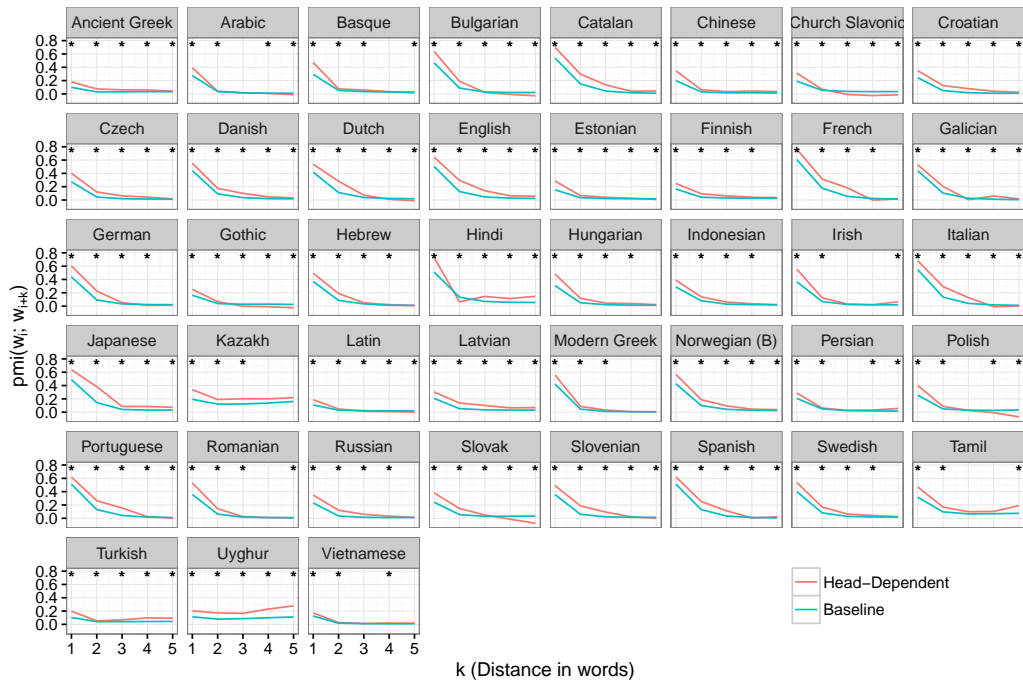


Figure 4: Average pointwise mutual information over POS tags for word pairs with k words intervening, for all words (baseline) and for words in a direct dependency relationship. Asterisks mark distances where the difference between the baseline and words in a dependency relationship is significant at $p < 0.005$ by Monte Carlo permutation tests over word pair observations with 500 samples.

The result of Gildea and Jaeger (2015), which shows that word orders in languages are optimized to minimize trigram surprisal of words, can be taken to show maximization of information locality under the noise distribution where context is truncated deterministically at length 2. Whereas Gildea and Jaeger (2015) treat dependency length minimization and trigram surprisal minimization as separate factors, under the view in this paper these two phenomena emerge as two aspects of information locality. In general, the mutual information of linguistic elements has been found to decrease with distance (Li, 1989; Lin and Tegmark, 2016), although this claim has only been tested for letters, not for larger linguistic units such as morphemes. The fact that linguistic units that are close typically have high mutual information could result from optimization of word order for information locality.

The idea that syntactically dependent words have high mutual information is also ubiquitously implicit in probabilistic models of language and in practical NLP models. For example, it is implied by head-outward generative models (Eisner, 1996; Eisner, 1997; Klein and Manning, 2004), the first successful models for grammar induction. Mutual information has been used directly for unsupervised discovery of syntactic dependencies (Yuret, 1998) and evaluation of dependency parses (de Paiva Alves, 1996), as well as commonly for collocation detection (Church and Hanks, 1990). In addition to providing evidence for a crucial assumption in the derivation of information locality, our results also give evidence backing up the theoretical validity of such models and methods.

The derivation of information locality given here assumed progressive erasure noise for concreteness, but we believe it should be possible to derive this generalization for a large family of noise distributions.

5 Conclusion

We have introduced a computational-level model of incremental sentence processing difficulty based on the principle that comprehenders have uncertainty about the previous input and act rationally on that uncertainty. Noisy-context surprisal accounts for key effects predicted by expectation-based and memory-based models, in addition to providing the first computational-level explanation of language-specific structural forgetting,

which involves subtle interactions between memory and probabilistic expectations. Noisy-context surprisal also leads to a general principle of information locality offering a new interpretation of syntactic locality effects, and leading to broader and potentially different predictions than purely memory-based models.

Here we have used qualitative arguments and have used different specific noise distributions to make different points. Our aim has been to argue for the theoretical viability of noisy-context surprisal, without committing the theory to a particular noise distribution. We believe our predictions will be derivable under very general classes of noise distributions, and we plan to pursue these more general derivations in future work.

A more psychologically accurate model will likely use a more nuanced noise distribution than the simple decay functions in this paper, which do not capture the subtleties of human memory. In particular, simple decay functions do not capture memory retrieval effects of the kind described in Anderson and Schooler (1991), where different items in a sequence have different propensities to be forgotten, in accordance with rational allocation of resources for retrieval. Seen as a noise distribution, this memory model implies that the erasure probability of a word is a function of the word's identity, and not only the word's position in the sequence as in Section 4.1.1. Including such noise distributions in the noisy-context surprisal model could provide a rich set of predictions to test the model more extensively.

Acknowledgments

We would like to thank members of Tedlab and the Computational Psycholinguistics Lab at MIT for helpful comments. R.F. was supported by NSF grant #1551543.

References

- John R. Anderson and Lael J. Schooler. 1991. Reflections of the environment in memory. *Psychological Science*, 2(6):396.
- Anthony J. Bell. 2003. The co-information lattice. In *Proceedings of the Fifth International Workshop on Independent Component Analysis and Blind Signal Separation*, pages 921–926.
- Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

- Eduardo de Paiva Alves. 1996. The selection of the most probable dependency structure in Japanese using mutual information. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 372–374.
- Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.
- Vera Demberg and Frank Keller. 2009. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*, Amsterdam, The Netherlands. Cognitive Science Society.
- Vera Demberg, Frank Keller, and Alexander Koller. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjointing grammar. *Computational Linguistics*, 39(4):1025–1066.
- Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 340–345.
- Jason M. Eisner. 1997. An empirical comparison of probability models for dependency grammar. Technical report, IRCS Report 96–11, University of Pennsylvania.
- Ramon Ferrer i Cancho. 2015. The placement of the head that minimizes online memory: a complex systems approach. *Language Dynamics and Change*, 5(1):114–137.
- Stefan L. Frank, Thijs Trompenaars, Richard L. Lewis, and Shravan Vasishth. 2016. Cross-linguistic differences in processing double-embedded relative clauses: Working-memory constraints or language statistics? *Cognitive Science*, 40:554–578.
- Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341.
- Edward Gibson and James Thomas. 1999. Memory limitations and structural forgetting: The perception of complex ungrammatical sentences as grammatical. *Language and Cognitive Processes*, 14(3):225–248.
- Edward Gibson, Steven T. Piantadosi, Kimberly Brink, Leon Bergen, Eunice Lim, and Rebecca Saxe. 2013. A noisy-channel account of crosslinguistic word-order variation. *Psychological science*, 24(7):1079–1088.
- Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.
- Daniel Gildea and T. Florian Jaeger. 2015. Human languages order information efficiently. *arXiv*, abs/1510.02823.
- Daniel Gildea and David Temperley. 2010. Do grammars minimize dependency length? *Cognitive Science*, 34(2):286–310.
- Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 241–247.
- Daniel Grodner and Edward Gibson. 2005. Consequences of the serial nature of linguistic input for sentential complexity. *Cognitive Science*, 29(2):261–290.
- John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics and Language Technologies*, pages 1–8.
- John A. Hawkins. 1994. *A performance theory of order and constituency*. Cambridge University Press, Cambridge.
- Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, page 478.
- Roger Levy. 2008a. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.
- Roger Levy. 2008b. A noisy-channel model of rational human sentence comprehension under uncertain input. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 234–243.
- Roger Levy. 2011. Integrating surprisal and uncertain-input models in online sentence comprehension: formal techniques and empirical results. In *ACL*, pages 1055–1065.
- Richard L. Lewis and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29(3):375–419.
- Wentian Li. 1989. Mutual information functions of natural language texts. Technical report, Santa Fe Institute Working Paper #1989-10-008.
- Henry W. Lin and Max Tegmark. 2016. Critical behavior from deep dynamics: A hidden dimension in natural language. *arXiv*, abs/1606.06737.
- David Marr. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman & Company.
- William J. McGill. 1955. Multivariate information transmission. *IEEE Transactions on Information Theory*, 4(4):93–111.

- Jeff Mitchell, Mirella Lapata, Vera Demberg, and Frank Keller. 2010. Syntactic and semantic factors in processing difficulty: An integrated measure. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 196–206.
- Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Eckhard Bick, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Drohanova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Claudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lucia Lam, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Robert Östling, Lilja Øvreliid, Valeria Paiva, Elena Pascual, Marco Passarotti, Cene-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Mats Wirén, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.
- Ting Qian and T. Florian Jaeger. 2012. Cue effectiveness in communicatively efficient discourse production. *Cognitive Science*, 36:1312–1336.
- Rajakrishnan Rajkumar, Marten van Schijndel, Michael White, and William Schuler. 2016. Investigating locality effects and surprisal in written English syntactic choice phenomena. *Cognition*, 155:204–232.
- Douglas Roland, Frederic Dick, and Jeffrey L. Elman. 2007. Frequency of basic English grammatical structures: A corpus analysis. *Journal of Memory and Language*, 57(3):348–379.
- Cory Shain, Marten van Schijndel, Richard Futrell, Edward Gibson, and William Schuler. 2016. Memory access during incremental sentence processing causes reading time latency. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CLALC)*, pages 49–58, Osaka, Japan.
- Nathaniel J. Smith and Roger Levy. 2013. The effect of word predictability on reading time is logarithmic. *Cognition*, 128(3):302–319.
- Shravan Vasishth, Katja Suckow, Richard L. Lewis, and Sabine Kern. 2010. Short-term forgetting in sentence comprehension: Crosslinguistic evidence from verb-final structures. *Language and Cognitive Processes*, 25(4):533–567.
- Deniz Yuret. 1998. Discovery of linguistic relations using lexical attraction. *arXiv preprint [cmp-lg/9805009](https://arxiv.org/abs/9805009)*.

Task-Specific Attentive Pooling of Phrase Alignments Contributes to Sentence Matching

Wenpeng Yin, Hinrich Schütze

The Center for Information and Language Processing
LMU Munich, Germany
wenpeng@cis.lmu.de

Abstract

This work studies comparatively two typical sentence matching tasks: textual entailment (TE) and answer selection (AS), observing that weaker phrase alignments are more critical in TE, while stronger phrase alignments deserve more attention in AS. The key to reach this observation lies in phrase detection, phrase representation, phrase alignment, and more importantly how to connect those aligned phrases of different matching degrees with the final classifier.

Prior work (i) has limitations in phrase generation and representation, or (ii) conducts alignment at word and phrase levels by handcrafted features or (iii) utilizes a single framework of alignment without considering the characteristics of specific tasks, which limits the framework’s effectiveness across tasks.

We propose an architecture based on Gated Recurrent Unit that supports (i) representation learning of phrases of *arbitrary granularity* and (ii) task-specific attentive pooling of phrase alignments between two sentences. Experimental results on TE and AS match our observation and show the effectiveness of our approach.

1 Introduction

How to model a pair of sentences is a critical issue in many NLP tasks, including textual entailment (Marelli et al., 2014a; Bowman et al., 2015a; Yin et al., 2016a) and answer selection (Yu et al., 2014; Yang et al., 2015; Santos et al., 2016). A key challenge common to these tasks is the lack of explicit alignment annotation between the sentences of the pair. Thus, inferring and assessing the semantic relations between words and phrases in the two sentences is a core issue.



Figure 1: Alignment examples in TE (top) and AS (bottom). Green color: identical (subset) alignment; blue color: relatedness alignment; red color: unrelated alignment. Q: the first sentence in TE or the question in AS; C^+ , C^- : the correct or incorrect counterpart in the sentence pair (Q , C).

Figure 1 shows examples of human annotated phrase alignments. In the TE example, we try to figure out Q entails C^+ (positive) or C^- (negative). As human beings, we discover the relationship of two sentences by studying the alignments between linguistic units. We see that some phrases are kept: “are playing outdoors” (between Q and C^+), “are playing ” (between Q and C^-). Some phrases are changed into related semantics on purpose: “the young boys” (Q) \rightarrow “the kids” (C^+ & C^-), “the man is smiling nearby” (Q) \rightarrow “near a man with a smile” (C^+) or \rightarrow “an old man is standing in the background” (C^-). We can see that the kept parts have stronger alignments (green color), and changed parts have weaker alignments (blue color). Here, by “strong” / “weak” we mean how semantically close the two aligned phrases are. To successfully identify the relationships of (Q , C^+) or (Q , C^-), studying the changed parts is crucial. Hence, we argue that TE should pay more attention to weaker alignments.

In AS, we try to figure out: does sentence C^+ or sentence C^- answer question Q ? Roughly, the content in candidates C^+ and C^- can be classified into aligned part (e.g., repeated or relevant parts) and negligible part. This differs from TE, in which it is hard to claim that some parts are negligible or play a minor role, as TE requires to make clear that each part can entail or be entailed. Hence, TE is considerably sensitive to those “unseen” parts. In contrast, *AS is more tolerant of negligible parts and less related parts*. From the AS example in Figure 1, we see that “Auburndale Florida” (Q) can find related part “the city” (C^+), and “Auburndale”, “a city” (C^-); “how big” (Q) also matches “had a population of 12,381” (C^+) very well. And some unaligned parts exist, denoted by red color. *Hence, we argue that stronger alignments in AS deserve more attention.*

The above analysis suggests that: (i) alignments connecting two sentences can happen between phrases of arbitrary granularity; (ii) phrase alignments can have different intensities; (iii) tasks of different properties require paying different attention to alignments of different intensities.

Alignments at word level (Yih et al., 2013) or phrase level (Yao et al., 2013) both have been studied before. For example, Yih et al. (2013) make use of WordNet (Miller, 1995) and Probase (Wu et al., 2012) for identifying hyper- and hyponymy. Yao et al. (2013) use POS tags, WordNet and paraphrase database for alignment identification. Their approaches rely on manual feature design and linguistic resources. We develop a deep neural network (DNN) to learn representations of phrases of arbitrary lengths. As a result, alignments can be searched in a more automatic and exhaustive way.

DNNs have been intensively investigated in sentence pair classifications (Blacoe and Lapata, 2012; Socher et al., 2011; Yin and Schütze, 2015b), and attention mechanisms are also applied to individual tasks (Santos et al., 2016; Rocktäschel et al., 2016; Wang and Jiang, 2016); however, most attention-based DNNs have implicit assumption that stronger alignments deserve more attention (Yin et al., 2016a; Santos et al., 2016; Yin et al., 2016b). Our examples in Figure 1, instead, show that this assumption does not hold invariably. Weaker alignments in certain tasks such as TE can be the indicator of the final decision. Our inspiration comes from the analysis of some prior work. For TE, Yin et al. (2016a)

show that considering the pairs in which overlapping tokens are removed can give a boost. This simple trick matches our motivation that weaker alignment should be given more attention in TE. However, Yin et al. (2016a) remove overlapping tokens completely, potentially obscuring complex alignment configurations. In addition, Yin et al. (2016a) use the same attention mechanism for TE and AS, which is less optimal based on our observations.

This motivates us in this work to introduce DNNs with a flexible attention mechanism that is adaptable for specific tasks. For TE, it can make our system pay more attention to weaker alignments; for AS, it enables our system to focus on stronger alignments. We can treat the pre-processing in (Yin et al., 2016a) as a hard way, and ours as a soft way, as our phrases have more flexible lengths and the existence of overlapping phrases decreases the risk of losing important alignments. In experiments, we will show that this attention scheme is very effective for different tasks.

We make the following contributions. (i) We use GRU (Gated Recurrent Unit (Cho et al., 2014)) to learn representations for phrases of arbitrary granularity. Based on phrase representations, we can detect phrase alignments of different intensities. (ii) We propose attentive pooling to achieve flexible choice among alignments, depending on the characteristics of the task. (iii) We achieve state-of-the-art on TE task.

2 Related Work

Non-DNN for sentence pair modeling. Heilman and Smith (2010) describe tree edit models that generalize tree edit distance by allowing operations that better account for complex re-ordering phenomena and by learning from data how different edits should affect the model’s decisions about sentence relations. Wang and Manning (2010) cope with the alignment between a sentence pair by using a probabilistic model that models tree-edit operations on dependency parse trees. Their model treats alignments as structured latent variables, and offers a principled framework for incorporating complex linguistic features. Guo and Diab (2012) identify the degree of sentence similarity by modeling the missing words (words that are not in the sentence) so as to relieve the sparseness issue of sentence modeling. Yih et

al. (2013) try to improve the shallow semantic component, lexical semantics, by formulating sentence pair as a semantic matching problem with a latent word-alignment structure as in (Chang et al., 2010). More fine-grained word overlap and alignment between two sentences are explored in (Lai and Hockenmaier, 2014), in which negation, hypernym/hyponym, synonym and antonym relations are used. Yao et al. (2013) extend word-to-word alignment to phrase-to-phrase alignment by a semi-Markov CRF. Such approaches often require more computational resources. In addition, using syntactic/semantic parsing during run-time to find the best matching between structured representation of sentences is not trivial.

DNN for sentence pair classification. There recently has been great interest in using DNNs for classifying sentence pairs as they can reduce the burden of feature engineering.

For TE, Bowman et al. (2015b) employ recursive DNN to encode entailment on SICK (Marelli et al., 2014b). Rocktäschel et al. (2016) present an attention-based LSTM (long short-term memory, Hochreiter and Schmidhuber (1997)) for the SNLI corpus (Bowman et al., 2015a).

For AS, Yu et al. (2014) present a bigram CNN (convolutional neural network (LeCun et al., 1998)) to model question and answer candidates. Yang et al. (2015) extend this method and get state-of-the-art performance on the WikiQA dataset. Feng et al. (2015) test various setups of a bi-CNN architecture on an insurance domain QA dataset. Tan et al. (2015) explore bidirectional LSTM on the same dataset. Other sentence matching tasks such as paraphrase identification (Socher et al., 2011; Yin and Schütze, 2015a), question – Freebase fact matching (Yin et al., 2016b) etc. are also investigated.

Some prior work aims to solve a general sentence matching problem. Hu et al. (2014) present two CNN architectures for paraphrasing, sentence completion (SC), tweet-response matching tasks. Yin and Schütze (2015b) propose the Multi-GranCNN architecture to model general sentence matching based on phrase matching on multiple levels of granularity. Wan et al. (2016) try to match two sentences in AS and SC by multiple sentence representations, each coming from the local representations of two LSTMs.

Attention-based DNN for alignment. DNNs have been successfully developed to detect align-

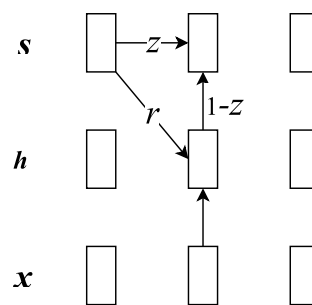


Figure 2: Gated Recurrent Unit

ments, e.g., in machine translation (Bahdanau et al., 2015; Luong et al., 2015) and text reconstruction (Li et al., 2015; Rush et al., 2015). In addition, attention-based alignment is also applied in natural language inference (e.g., Rocktäschel et al. (2016), Wang and Jiang (2016)). However, most of this work aligns word-by-word. As Figure 1 shows, many sentence relations can be better identified through phrase level alignments. This is one motivation of our work.

3 Model

This section first gives a brief introduction of GRU and how it performs phrase representation learning, then describes the different attentive poolings for phrase alignments w.r.t TE and AS tasks.

3.1 GRU Introduction

GRU is a simplified version of LSTM. Both are found effective in sequence modeling, as they are order-sensitive and can capture long-range context. The tradeoffs between GRU and its competitor LSTM have not been fully explored yet. According to empirical evaluations in (Chung et al., 2014; Jozefowicz et al., 2015), there is not a clear winner. In many tasks both architectures yield comparable performance and tuning hyper-parameters like layer size is probably more important than picking the ideal architecture. GRU have fewer parameters and thus may train a bit faster or need less data to generalize. Hence, we use GRU, as shown in Figure 2, to model text:

$$\mathbf{z} = \sigma(\mathbf{x}_t \mathbf{U}^z + \mathbf{s}_{t-1} \mathbf{W}^z) \quad (1)$$

$$\mathbf{r} = \sigma(\mathbf{x}_t \mathbf{U}^r + \mathbf{s}_{t-1} \mathbf{W}^r) \quad (2)$$

$$\mathbf{h}_t = \tanh(\mathbf{x}_t \mathbf{U}^h + (\mathbf{s}_{t-1} \circ \mathbf{r}) \mathbf{W}^h) \quad (3)$$

$$\mathbf{s}_t = (1 - \mathbf{z}) \circ \mathbf{h}_t + \mathbf{z} \circ \mathbf{s}_{t-1} \quad (4)$$

x is the input sentence with token $\mathbf{x}_t \in \mathbb{R}^d$ at position t , $\mathbf{s}_t \in \mathbb{R}^h$ is the hidden state at t , supposed to

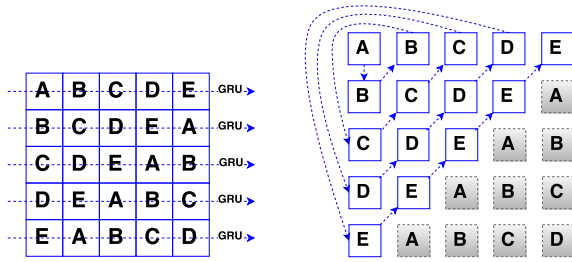


Figure 3: Phrase representation learning by GRU (left), sentence reformatting (right)

encode the history x_1, \dots, x_{t-1} . \mathbf{z} and \mathbf{r} are two gates. All $\mathbf{U} \in \mathbb{R}^{d \times h}$, $\mathbf{W} \in \mathbb{R}^{h \times h}$ are parameters in GRU.

3.2 Representation Learning for Phrases

For a general sentence s with five consecutive words: ABCDE, with each word represented by a word embedding of dimensionality d , we first create four fake sentences, s^1 : “BCDEA”, s^2 : “CDEAB”, s^3 : “DEABC” and s^4 : “EABCD”, then put them in a matrix (Figure 3, left).

We run GRUs on each row of this matrix in parallel. As GRU is able to encode the whole sequence up to current position, this step generates representations for any consecutive phrases in original sentence s . For example, the GRU hidden state at position “E” at coordinates (1,5) (i.e., 1st row, 5th column) denotes the representation of the phrase “ABCDE” which in fact is s itself, the hidden state at “E” (2,4) denotes the representation of phrase “BCDE”, . . . , the hidden state of “E” (5,1) denotes phrase representation of “E” itself. Hence, for each token, we can learn the representations for all phrases ending with this token. Finally, all phrases of any lengths in s can get a representation vector. GRUs in those rows are set to share weights so that all phrase representations are comparable in the same space.

Now, we reformat sentence “ABCDE” into $s^* =$ “(A) (B) (AB) (C) (BC) (ABC) (D) (CD) (BCD) (ABCD) (E) (DE) (CDE) (BCDE) (ABCDE)”, as shown by arrows in Figure 3 (right), the arrow direction means phrase order. Each sequence in parentheses is a phrase (we use parentheses just for making the phrase boundaries clear). Randomly taking a phrase “CDE” as an example, its representation comes from the hidden state at “E” (3,3) in Figure 3 (left). Shaded parts are discarded. The main advantage of reformatting sentence “ABCDE” into the *new sentence* s^* is to cre-

ate phrase-level semantic units, but at the same time we maintain the order information.

Hence, the sentence “how big is Auburndale Florida” in Figure 1 will be reformatted into “(how) (big) (how big) (is) (big is) (how big is) (Auburndale) (is Auburndale) (big is Auburndale) (how big is Auburndale) (Florida) (Auburndale Florida) (is Auburndale Florida) (big is Auburndale Florida) (how big is Auburndale Florida)”. We can see that phrases are exhaustively detected and represented.

In the experiments of this work, we explore the phrases of maximal length 7 instead of arbitrary lengths.

3.3 Attentive Pooling

As each sentence s^* consists of a sequence of phrases, and each phrase is denoted by a representation vector generated by GRU, we can compute an *alignment matrix* \mathbf{A} between two sentences s_1^* and s_2^* , by comparing each two phrases, one from s_1^* and one from s_2^* . Let s_1^* and s_2^* also denote lengths respectively, thus $\mathbf{A} \in \mathbb{R}^{s_1^* \times s_2^*}$. While there are many ways of computing the entries of \mathbf{A} , we found that cosine works well in our setting.

The first step then is to detect the best alignment for each phrase by leveraging \mathbf{A} . To be concrete, for sentence s_1^* , we do row-wise max-pooling over \mathbf{A} as attention vector \mathbf{a}_1 :

$$\mathbf{a}_{1,i} = \max(\mathbf{A}[i, :]) \quad (5)$$

In \mathbf{a}_1 , the entry $\mathbf{a}_{1,i}$ denotes the best alignment for i^{th} phrase in sentence s_1^* . Similarly, we can do column-wise max-pooling to generate attention vector \mathbf{a}_2 for sentence s_2^* .

Now, the problem is that we need to pay most attention to the phrases aligned very well or phrases aligned badly. According to the analysis of the two examples in Figure 1, we need to pay more attention to weaker (resp. stronger) alignments in TE (resp. AS). To this end, we adopt different second step over attention vector \mathbf{a}_i ($i = 1, 2$) for TE and AS.

For TE, in which weaker alignments are supposed to contribute more, we do *k-min-pooling* over \mathbf{a}_i , i.e., we only keep the k phrases which are aligned worst. For the (Q, C^+) pair in TE example of Figure 1, we expect this step is able to put most of our attention to the phrases “the kids”, “the young boys”, “near a man with a smile” and “and the man is smiling nearby” as they have rela-

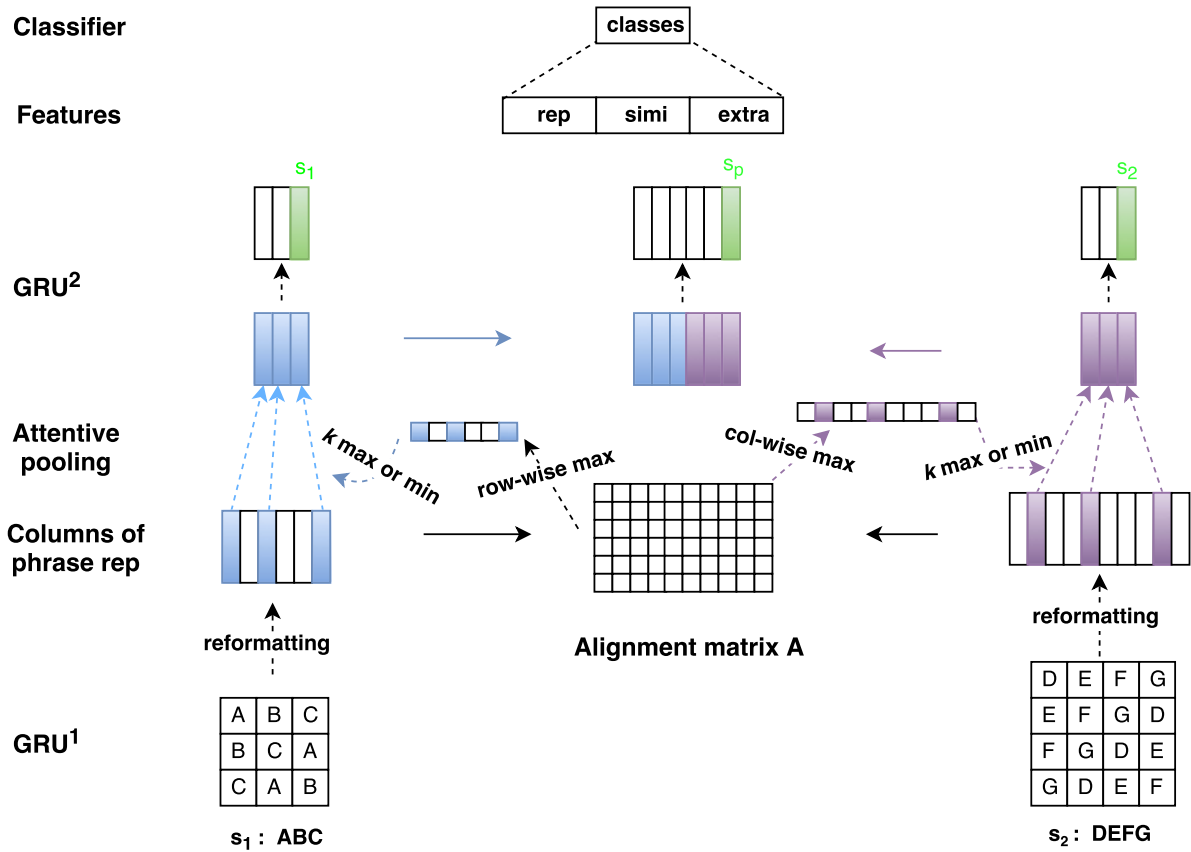


Figure 4: The whole architecture

tively weaker alignments while their relations are the indicator of the final decision.

For AS, in which stronger alignments are supposed to contribute more, we do k -max-pooling over \mathbf{a}_i , i.e., we only keep the k phrases which are aligned best. For the (Q, C^+) pair in AS example of Figure 1, we expect this k -max-pooling is able to put most of our attention to the phrases “how big” “Auburndale Florida”, “the city” and “had a population of 12,381” as they have relatively stronger alignments and their relations are the indicator of the final decision. We keep the original order of extracted phrases after k -min/max-pooling.

In summary, for TE, we first do row-wise max-pooling over alignment matrix, then do k -min-pooling over generated alignment vector; we use k -min-max-pooling to denote the whole process. In contrast, we use k -max-max-pooling for AS. We refer to this method of using two successive min or max pooling steps as *attentive pooling*.

3.4 The Whole Architecture

Now, we present the whole system in Figure 4. We take sentences s_1 “ABC” and s_2 “DEFG” as illustration. Each token, i.e., A to F, in the figure is denoted by an embedding vector, hence each sentence is represented as an order-3 tensor as input (they are depicted as rectangles just for simplicity). Based on tensor-style sentence input, we have described the phrase representation learning by GRU¹ in Section 3.2 and attentive pooling in Section 3.3.

Attentive pooling generates a new feature map for each sentence, as shown in Figure 4 (the third layer from the bottom), and each column representation in the feature map denotes a key phrase in this sentence that, based on our modeling assumptions, should be a good basis for the correct final decision. For instance, we expect such a feature map to contain representations of “the young boys”, “outdoors” and “and the man is smiling nearby” for the sentence Q in the TE example of Figure 1.

Now, we do another GRU² step for: 1) the new

	d	lr	bs	L_2	div	k
TE	[256,256]	.0001	1	.0006	.06	5
AS	[50,50]	.0001	1	.0006	.06	6

Table 1: Hyperparameters. d : dimensionality of hidden states in GRU layers; lr: learning rate; bs: mini-batch size; L_2 : L_2 normalization; div : diversity regularizer; k : k -min/max-pooling.

feature map of each sentence mentioned above, to encode all the key phrases as the sentence representation; 2) a concatenated feature map of the two new sentence feature maps, to encode all the key phrases in the two sentences sequentially as the representation of the *sentence pair*. As GRU generates a hidden state at each position, we always choose the last hidden state as the representation of the sentence or sentence pair. In Figure 4 (the fourth layer), these final GRU-generated representations for sentence s_1 , s_2 and the sentence pair are depicted as green columns: s_1 , s_2 and s_p respectively.

As for the input of the final classifier, it can be flexible, such as representation vectors (*rep*), similarity scores between s_1 and s_2 (*simi*), and extra linguistic features (*extra*). This can vary based on the specific tasks. We give details in Section 4.

4 Experiments

We test the proposed architectures on TE and AS benchmark datasets.

4.1 Common Setup

For both TE and AS, words are initialized by 300-dimensional GloVe embeddings¹ (Pennington et al., 2014) and not changed during training. A single randomly initialized embedding is created for all unknown words by uniform sampling from $[-.01, .01]$. We use ADAM (Kingma and Ba, 2015), with a first momentum coefficient of 0.9 and a second momentum coefficient of 0.999,² L_2 regularization and Diversity Regularization (Xie et al., 2015). Table 1 shows the values of the hyperparameters, tuned on dev.

Classifier. Following Yin et al. (2016a), we use three classifiers – logistic regression in DNN, logistic regression and linear SVM with default parameters³ directly on the feature vector – and report performance of the best.

¹nlp.stanford.edu/projects/glove/

²Standard configuration recommended by Kingma and Ba

³<http://scikit-learn.org/stable/> for both.

Common Baselines. (i) **Addition.** We sum up word embeddings element-wise to form sentence representation, then concatenate two sentence representation vectors (s_1^0, s_2^0) as classifier input. (ii) **A-LSTM.** The pioneering attention based LSTM system for a specific sentence pair classification task “natural language inference” (Rocktäschel et al., 2016). A-LSTM has the same dimensionality as our GRU system in terms of initialized word representations and the hidden states. (iii) **ABCNN** (Yin et al., 2016a). The state-of-the-art system in both TE and AS.

Based on the motivation in Section 1, the main hypothesis to be tested in experiments is: k -min-max-pooling is superior for TE and k -max-max-pooling is superior for AS. In addition, we would like to determine whether the second pooling step in attention pooling, i.e., the k -min/max-pooling, is more effective than a “full-pooling” in which *all* the generated phrases are forwarded into the next layer.

4.2 Textual Entailment

SemEval 2014 Task 1 (Marelli et al., 2014a) evaluates system predictions of textual entailment (TE) relations on sentence pairs from the SICK dataset (Marelli et al., 2014b). The three classes are entailment, contradiction and neutral. The sizes of SICK train, dev and test sets are 4439, 495 and 4906 pairs, respectively. *We choose SICK benchmark dataset so that our result is directly comparable with that of (Yin et al., 2016a), in which non-overlapping text are utilized explicitly to boost the performance. That trick inspires this work.*

Following Lai and Hockenmaier (2014), we train our final system (after fixing of hyperparameters) on train and dev (4,934 pairs). Our evaluation measure is accuracy.

4.2.1 Feature Vector

The final feature vector as input of classifier contains three parts: *rep*, *simi*, *extra*.

Rep. Totally five vectors, three are the top sentence representation s_1, s_2 and the top sentence pair representation s_p (shown in green in Figure 4), two are s_1^0, s_2^0 from *Addition* baseline.

Simi. Four similarity scores, cosine similarity and euclidean distance between s_1 and s_2 , cosine similarity and euclidean distance between s_1^0 and s_2^0 . Euclidean distance $\| \cdot \|$ is transformed into $1/(1 + \| \cdot \|)$.

	method	acc
SemEval Top3	(Jimenez et al., 2014)	83.1
	(Zhao et al., 2014)	83.6
	(Lai and Hockenmaier, 2014)	84.6
TrRNTN	(Bowman et al., 2015b)	76.9
Addition	no features	73.1
	plus features	79.4
A-LSTM	no features	78.0
	plus features	81.7
ABCNN	(Yin et al., 2016a)	86.2
GRU <i>k</i> -min-max ablation	– rep	86.4
	– simi	85.1
	– extra	85.5
GRU	<i>k</i> -max-max-pooling	84.9
	full-pooling	85.2
	<i>k</i> -min-max-pooling	87.1*

Table 2: Results on SICK. Significant improvement over both *k*-max-max-pooling and full-pooling is marked with * (test of equal proportions, $p < .05$).

Extra. We include the same 22 linguistic features as Yin et al. (2016a). They cover 15 machine translation metrics between the two sentences; whether or not the two sentences contain negation tokens like “no”, “not” etc; whether or not they contain synonyms, hypernyms or antonyms; two sentence lengths. See Yin et al. (2016a) for details.

4.2.2 Results

Table 2 shows that GRU with *k*-min-max-pooling gets state-of-the-art performance on SICK and significantly outperforms *k*-max-max-pooling and full-pooling. Full-pooling has more phrase input than the combination of *k*-max-max-pooling and *k*-min-max-pooling, this might bring two problems: (i) noisy alignments increase; (ii) sentence pair representation s_p is no longer discriminative – s_p does not know its semantics comes from phrases of s_1 or s_2 : as different sentences have different lengths, the boundary location separating two sentences varies across pairs. However, this is crucial to determine whether s_1 entails s_2 .

ABCNN (Yin et al., 2016a) is based on assumptions similar to *k*-max-max-pooling: words/phrases with higher matching values should contribute more in this task. However, ABCNN gets the optimal performance by combining a reformatted SICK version in which

	method	MAP	MRR
Baselines	CNN-Cnt	0.6520	0.6652
	Addition	0.5021	0.5069
	Addition-Cnt	0.5888	0.5929
	A-LSTM	0.5321	0.5469
	A-LSTM-Cnt	0.6388	0.6529
	AP-CNN	0.6886	0.6957
GRU <i>k</i> -max-max ablation	ABCNN	0.6921	0.7127
	– rep	0.6913	0.6994
	– simi	0.6764	0.6875
GRU	– extra	0.6802	0.6899
	<i>k</i> -min-max-pooling	0.6674	0.6791
	full-pooling	0.6693	0.6785
	<i>k</i> -max-max-pooling	0.7124*	0.7237*

Table 3: Results on WikiQA. Significant improvement over both *k*-min-max-pooling and full-pooling is marked with * (t -test, $p < .05$). STOA: 74.17 (MAP)/75.88 (MRR) in (Tymoshenko et al., 2016)

overlapping tokens in two sentences are removed. This instead hints that non-overlapping units can do a big favor for this task, which is indeed the superiority of our “*k*-min-max-pooling”.

4.3 Answer Selection

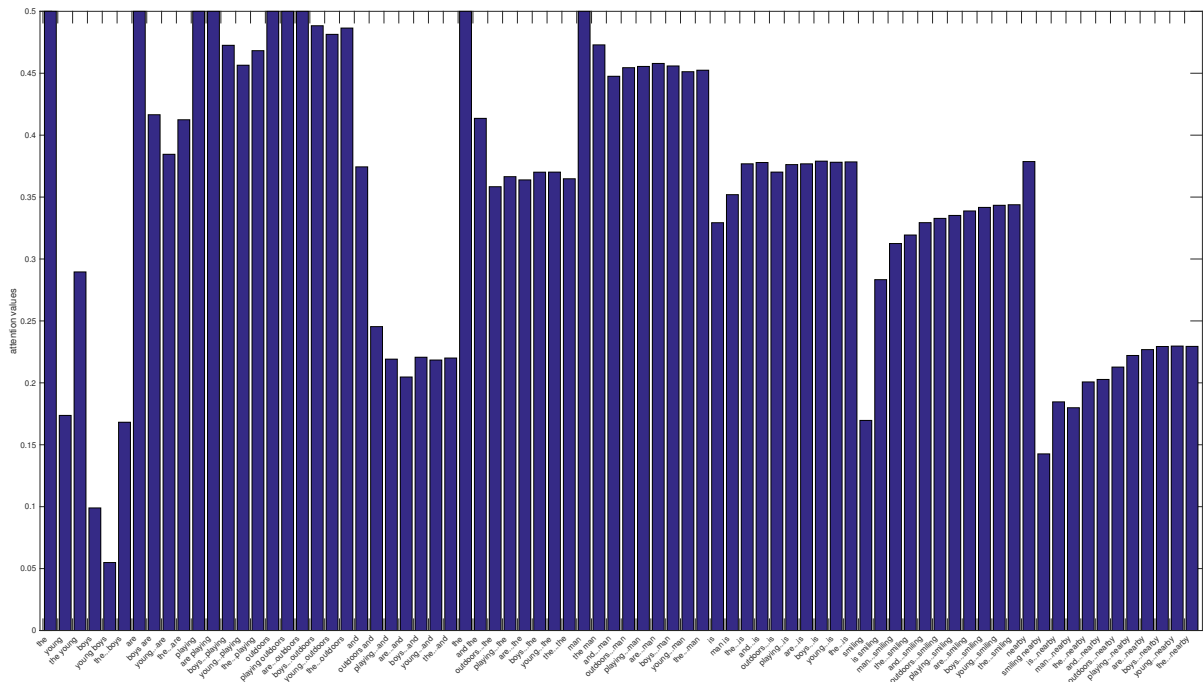
We use WikiQA⁴ subtask that assumes there is at least one correct answer for a question. This dataset consists of 20,360, 1130 and 2352 question-candidate pairs in train, dev and test, respectively. Following Yang et al. (2015), we truncate answers to 40 tokens and report mean average precision (MAP) and mean reciprocal rank (MRR).

Apart from the common baselines Addition, A-LSTM and ABCNN, we compare further with: (i) **CNN-Cnt** (Yang et al., 2015): combine CNN with two linguistic features “WordCnt” (the number of non-stopwords in the question that also occur in the answer) and “WgtWordCnt” (reweight the counts by the IDF values of the question words); (ii) **AP-CNN** (Santos et al., 2016).

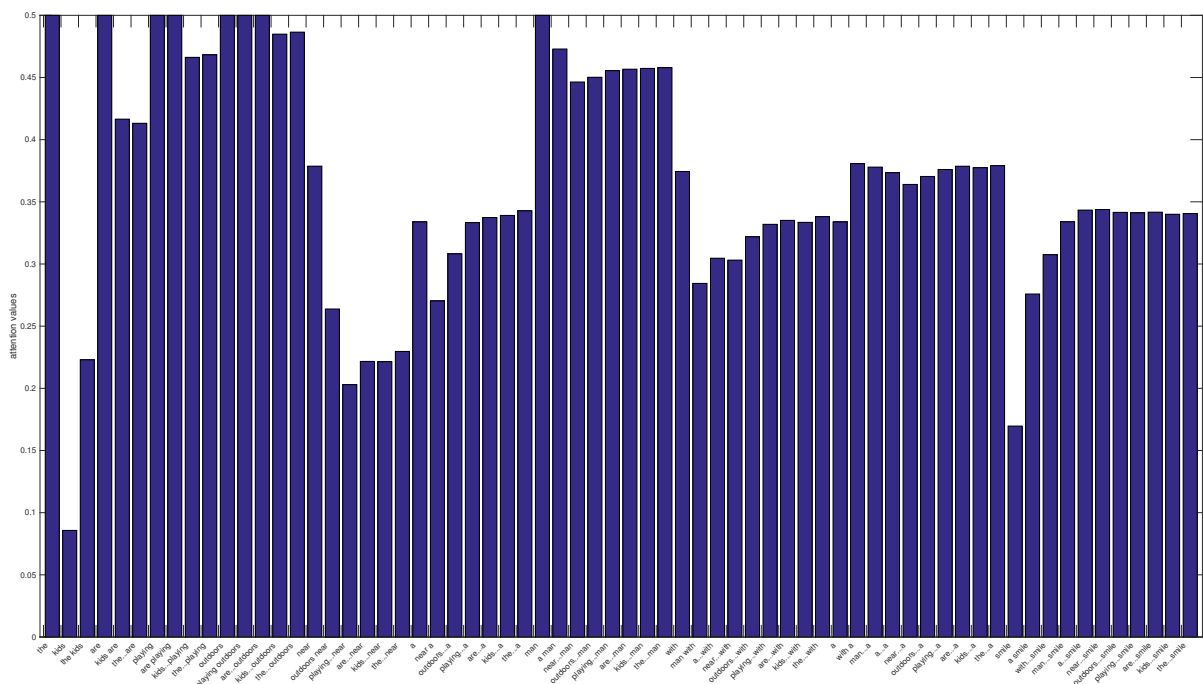
4.3.1 Feature Vector

The final feature vector in AS has the same (*rep*, *simi*, *extra*) structure as TE, except that **simi** consists of only two cosine similarity scores, and **extra** consists of four entries: two sentence lengths, WordCnt and WgtWordCnt.

⁴<http://aka.ms/WikiQA> (Yang et al., 2015)



(a) Attention distribution for phrases in "Q" of TE example in Figure 1



(b) Attention distribution for phrases in "C+" of TE example in Figure 1

Figure 5: Attention Visualization

4.3.2 Results

Table 3 shows that GRU with k -max-max-pooling is significantly better than its k -min-max-pooling and full-pooling versions. GRU with k -max-max-pooling has similar assumption with ABCNN (Yin et al., 2016a) and AP-CNN (Santos et al., 2016): units with higher matching scores are supposed to contribute more in this task. Our improvement

can be due to that: i) our linguistic units cover more exhaustive phrases, it enables alignments in a wider range; ii) we have two max-pooling steps in our attention pooling, especially the second one is able to remove some noisily aligned phrases. Both ABCNN and AP-CNN are based on convolutional layers, the phrase detection is constrained by filter sizes. Even though ABCNN tries a second

CNN layer to detect bigger-granular phrases, their phrases in different CNN layers cannot be aligned directly as they are in different spaces. GRU in this work uses the same weights to learn representations of arbitrary-granular phrases, hence, all phrases can share the representations in the same space and can be compared directly.

4.4 Visual Analysis

In this subsection, we visualize the attention distributions over phrases, i.e., a_i in Equation 5, of example sentences in Figure 1 (for space limit, we only show this for TE example). Figures 5(a)-5(b) respectively show the attention values of each phrase in (Q, C^+) pair in TE example in Figure 1. We can find that k -min-pooling over this distributions can indeed detect some key phrases that are supposed to determine the pair relations. Taking Figure 5(a) as an example, phrases “young boys”, phrases ending with “and”, phrases “smiling”, “is smiling”, “nearby” and a couple of phrases ending with “nearby” have lowest attention values. According to our k -min-pooling step, these phrases will be detected as key phrases. Considering further the Figure 5(b), phrases “kids”, phrases ending with “near”, and a couple of phrases ending with “smile” are detected as key phrases.

If we look at the key phrases in both sentences, we can find that the discovering of those key phrases matches our analysis in Section 1 for TE example: “kids” corresponds to “young boys”, “smiling nearby” corresponds to “near...smile”.

Another interesting phenomenon is that, taking Figure 5(b) as example, even though “are playing outdoors” can be well aligned as it appears in both sentences, nevertheless the visualization figures show that the attention values of “are playing outdoors and” in Q and “are playing outdoors near” drop dramatically. This hints that our model can get rid of some surface matching, as the key token “and” or “near” makes the semantics of “are playing outdoors and” and “are playing outdoors near” be pretty different with their sub-phrase “are playing outdoors”. This is important as “and” or “near” is crucial unit to connect the following key phrases “smiling nearby” in Q or “a smile” in C^+ . If we connect those key phrases sequentially as a new fake sentence, as we did in attentive pooling layer of Figure 4, we can see that the fake sentence roughly “reconstructs” the meaning of the original sentence while it is composed of phrase-level se-

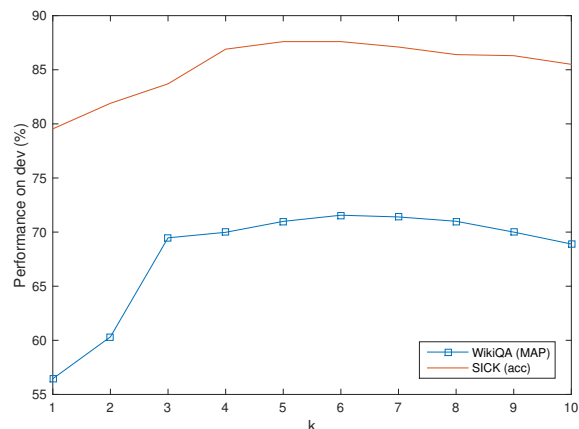


Figure 6: Effects of pooling size k (cf. Table Table 1)

mantic units now.

4.5 Effects of Pooling Size k

The key idea of the proposed method is achieved by the k -min/max pooling. We show how the hyperparameter k influences the results by tuning on the dev sets.

In Figure 6, we can see the performance trends of changing k value between 1 and 10 in the two tasks. Roughly $k > 4$ can give competitive results, but larger values bring performance drop.

5 Conclusion

In this work, we investigate the contribution of different intensities of phrase alignments for different tasks. We argue that it is not true that stronger alignments always matter more. We found TE task prefers weaker alignments while AS task prefers stronger alignments. We proposed flexible attentive poolings in GRU system to satisfy the different requirements of different tasks. Experimental results show the soundness of our argument and the effectiveness of our attention pooling based GRU systems.

As future work, we plan to investigate phrase representation learning in context and how to conduct the attentive pooling automatically regardless of the categories of the tasks.

Acknowledgments

We gratefully acknowledge the support of Deutsche Forschungsgemeinschaft for this work (SCHU 2246/8-2).

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.
- William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*, pages 546–556.
- Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.
- Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of CVSC workshop*, pages 12–21.
- Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of NAACL-HLT*, pages 429–437.
- Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *Proceedings of IEEE ASRU Workshop*.
- Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of ACL*, pages 864–872.
- Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*, pages 1011–1019.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*, pages 2042–2050.
- Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. *SemEval*, pages 732–742.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*, pages 2342–2350.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.
- Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. *SemEval*, pages 329–334.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*, pages 1106–1115.
- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.
- Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval*, pages 1–8.
- Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223.
- George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR*.
- Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389.
- Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.
- Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809.

- Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.
- Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL-HLT*, pages 1268–1278.
- Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of AAAI*, pages 2835–2841.
- Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of NAACL*, pages 1442–1451.
- Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of Coling*, pages 1164–1172.
- Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of SIGMOD*, pages 481–492.
- Pengtao Xie, Yuntian Deng, and Eric Xing. 2015. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*.
- Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018.
- Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Semi-markov phrase-based monolingual alignment. In *Proceedings of EMNLP*, pages 590–600.
- Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*, pages 1744–1753.
- Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proceedings of NAACL*, pages 901–911, May–June.
- Wenpeng Yin and Hinrich Schütze. 2015b. Multi-granccnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of ACL-IJCNLP*, pages 63–73.
- Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016a. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.
- Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016b. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING*, pages 1746–1756.
- Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS Deep Learning Workshop*.
- Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *SemEval*, pages 271–277.

On-demand Injection of Lexical Knowledge for Recognising Textual Entailment

Pascual Martínez-Gómez¹ Koji Mineshima²
pascual.mg@aist.go.jp mineshima.koji@ocha.ac.jp

Yusuke Miyao^{1,3,4} Daisuke Bekki^{1,2,3}
yusuke@nii.ac.jp bekki@is.ocha.ac.jp

¹Artificial Intelligence Research Center, AIST

²Ochanomizu University

³National Institute of Informatics and JST, PRESTO

⁴The Graduate University for Advanced Studies (SOKENDAI)
Tokyo, Japan

Abstract

We approach the recognition of textual entailment using logical semantic representations and a theorem prover. In this setup, lexical divergences that preserve semantic entailment between the source and target texts need to be explicitly stated. However, recognising subsentential semantic relations is not trivial. We address this problem by monitoring the proof of the theorem and detecting unprovable sub-goals that share predicate arguments with logical premises. If a linguistic relation exists, then an appropriate axiom is constructed on-demand and the theorem proving continues. Experiments show that this approach is effective and precise, producing a system that outperforms other logic-based systems and is competitive with state-of-the-art statistical methods.

1 Introduction

Recognising Textual Entailment (RTE) is a challenging NLP application where the objective is to judge whether a text fragment H logically follows from another text fragment T (Dagan et al., 2013). Advances in RTE have potentially positive implications in other areas such as fact checking, question-answering or information retrieval. Solutions to the RTE problem span a wide array of methods. Some methods are purely statistical (Lai and Hockenmaier, 2014; Zhao et al., 2014), where a classifying function is estimated using lexical or syntactic features. Other methods are purely se-

mantic (Bos et al., 2004), where logical formulas that represent the text fragments are constructed and used in a formal proof system. And yet others are hybrid systems (Beltagy et al., 2013), where a combination of statistical features and logical formulas are used to judge entailment relations.

In this paper, we adopt a strategy based on logics, encouraged by the high-performance that these systems achieve in linguistically challenging datasets (Abzianidze, 2015; Mineshima et al., 2015). An important advantage of these systems (including ours) is that they are unsupervised, thus no training data is necessary and no parameters need to be adjusted.

Under the perspective of these logic-based systems, there are mainly two associated challenges when solving RTE problems. The first challenge is to model the logics of the language with the purpose to represent the semantics of text fragments accurately. To this end, we follow the standard practice in formal semantics where the meaning of sentences is represented using logical formulas. The second challenge is to account for lexical relations between text fragments, typically between words or non-compositional phrases. We dedicate our efforts to the latter challenge, and assume that wide coverage linguistic resources are available to signal potential relations between lexical items in text fragments. The question is then how to make the best use of these linguistic resources to close the lexical gap between source and target text fragments.

Our contribution is a precise mechanism that allows to construct and use linguistic axioms on-demand. This mechanism monitors the progress

of a logical proof, detects unprovable sub-goals, and inserts axioms when necessary if a lexical relation is found in an external linguistic resource. These linguistic axioms encode lexical relations between specific segments of the source and target text fragments, thus accounting for lexical divergences that preserve semantic inclusion. To the best of our knowledge, this is the first attempt to integrate on-demand axiom injection into a purely logical natural deduction proof to recognise textual entailment. In the SICK dataset, our system obtains the highest accuracy among the logic systems, and competitive results with respect to machine learning approaches. We believe that our formulation is general enough to be extended to other semantic systems and to introduce lexical knowledge from ontological resources or statistical classifiers efficiently and effectively.

2 Related Work

Our work on recognising textual entailment is primarily inspired by Bos and Markert (2005), where first-order logic interpretations of sentences are used to prove entailment relations with theorem provers and model builders. These semantic interpretations were composed using Boxer (Bos et al., 2004) from derivations of a Combinatory Categorical Grammar (CCG) (Steedman, 2000) automatically obtained by C&C, a wide-coverage CCG parser (Clark and Curran, 2007). This system was later extended into Nutcracker (Bjerva et al., 2014), where WordNet (Miller, 1995) and relations from Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) are used to introduce external linguistic resources to account for lexical divergences (Pavlick et al., 2015). Pavlick et al. (2015) study the characteristics of linguistic relations that may signal entailment or contradiction at sub-sentential level. However, they ignore the logical context in which these linguistic relations occur in the entailment problem. Moreover, Nutcracker is not a purely logical system in that it uses a proof-approximation method with model-builders.

By contrast, our system is purely logic-based, in that it solely relies on proof constructions based on natural deduction system to make entailment judgements. In addition, as we will see below, a goal-directed proof construction procedure in our system is naturally combined with on-demand axiom injection, as opposed to simply selecting any two arbitrary phrases from T and H that display

any linguistic relation.

Beltagy et al. (2013) also use Boxer for their logical semantic representations but assign distributional similarity scores to any two phrases from T and H on-the-fly. Their approach is different from ours in that they use probabilistic logic as an underlying logic. Furthermore, the method to create relevant axioms in Beltagy et al. (2013) is based on a naïve enumeration, and they ignore the logical clues on when two phrases are candidates to be related, which we argue against.

Abzianidze (2015) presents a purely logic-based RTE system that uses CCG parsers and a natural-logic-based tableaux prover. However, his logical representations are based on a non-standard natural logic, which requires the definition of new inference rules for each logical word (e.g. *every*, *some*, *no*) and for which generic theorem provers are not reusable. Regarding the introduction of linguistic knowledge, the author uses only WordNet. However, during the learning phase, he adds missing knowledge manually (e.g. *note* is a hyponym of *paper*), whereas we restrict our results to those automatically generated.

Perhaps the most similar strategies to ours are those of Tian et al. (2014) and Beltagy et al. (2016), where the authors produce on-the-fly knowledge when the hypothesis H cannot be proved. In the work of Tian et al. (2014), propositions between T and H are aligned using logical clues; then, dependency paths are extracted between these propositions and WordNet or word vectors are used to assess the similarity between paths. However, the expressive power of their underlying representation system in Dependency-based Compositional Semantics (DCS) is rather limited and much weaker than the full first-order logic (Liang et al., 2013). Several extensions have been proposed (Tian et al., 2014; Dong et al., 2014), yet these DCS-based inference systems are a non-standard axiomatic system with many axioms and tend to be ad hoc. Whereas their semantic representations are specific to their logic framework, ours are well-understood, logically transparent representations that are generic to most state-of-the-art theorem provers using first-order logic.

Beltagy et al. (2016) use a Modified Robinson Resolution strategy to align clauses and literals between T and H . These alignments also constrain how the unaligned fragments of T and H may correspond to each other, reducing the

problem to a word or phrasal entailment recognition using a statistical classifier. However, that work only considers one possible set of alignments between T - H fragments, which has a decaying coverage when there is repetitions of content words and meta-predicates (typically occurring in medium and long sentences). Instead, we consider multiple alignments by backtracking the decisions on variable and predicate unifications, which is a more powerful strategy. Beltagy et al. (2016) use Markov Logic Networks (MLNs), which is an elegant framework that combines logics and probabilistic reasoning. However, the construction of their Markov Networks is limited by first-order logic, which may pose problems to represent modality or generalised quantifiers. Instead, our logical representations can also be used in a more expressive, higher-order inference system such as the one in Martínez-Gómez et al. (2016), as it was shown by Mineshima et al. (2015) and Mineshima et al. (2016) in a practical application for RTE.

3 Background

This section provides some basic background on our logic-based approach to Recognising Textual Entailment (RTE). RTE is a task of determining whether or not a given text (T) entails a given hypothesis (H). In logic-based approaches, T and H are mapped onto logical formulas; whether T entails H is then determined by checking whether $T \rightarrow H$ is a theorem in a logical system, possibly with the help of a knowledge base.

To obtain logical formulas for input sentences, we use the framework of Combinatory Categorical Grammar (CCG) (Steedman, 2000), a lexicalized grammar formalism that provides a transparent interface between syntax and semantics. We follow the standard method of building compositional semantics in CCG-based systems (Blackburn and Bos, 2005; Bos, 2008), where each syntactic category is schematically assigned a meaning representation formally specified as a λ -term. By combining the meanings of constituent words that appear in a CCG derivation tree, we can obtain a logical formula that serves as a semantic representation of an input sentence.

For semantic representations, we adopt Neo-Davidsonian Event Semantics (Parsons, 1990; Bos, 2008; Jurafsky and Martin, 2009). For instance, the sentence in (1) is mapped not to a sim-

ple formula (2) but to a formula (3) that involves an event variable.

- (1) John greets Mary.
- (2) $\text{greet}(\text{john}, \text{mary})$
- (3) $\exists v(\text{greet}(v) \wedge (\text{Subj}(v) = \text{john}) \wedge (\text{Obj}(v) = \text{mary}))$

The sentence (3) expresses that there is an event of greeting such that its subject is John and its object is Mary. In our Neo-Davidsonian approach, every verb is decomposed into a one-place predicate over events and a set of functional expressions such as $\text{Subj}(v) = \text{john}$, which relates an event to its participant.

VP-modifiers such as adverbs and prepositional phrases are also analysed as event predicates. For instance, (4) and (5) are analysed as having the semantic representations in (6) and (7), respectively.

- (4) John greets Mary warmly.
- (5) John walks to a station.
- (6) $\exists v(\text{greet}(v) \wedge (\text{Subj}(v) = \text{john}) \wedge (\text{Obj}(v) = \text{mary}) \wedge \text{warmly}(v))$
- (7) $\exists v(\text{walk}(v) \wedge (\text{Subj}(v) = \text{john}) \wedge \exists x(\text{station}(x) \wedge (\text{Goal}(v) = x)))$

There are several advantages of using event semantic formulas as representations for natural language inferences. First, it logically derives an entailment pattern to drop adverbial modifiers, such as the one from (4) to (1) and the one from (5) to *John walks*. Another advantage over simple representations like (2) is that it provides a uniform way of capturing the lexical relationship between verbs. For instance, the hypernym relation between the transitive verb *greet* and the intransitive verb *move* is represented as a simple axiom $\forall v(\text{greet}(v) \rightarrow \text{move}(v))$. This is possible because both verbs are analysed as one-place predicates over events, rather than as predicates with different arities such as $\text{greet}(x, y)$ and $\text{move}(x)$. All these inferences are derivable using the standard first-order logic. For these reasons, event semantic formulas are suitable for the purpose of performing logical inferences with lexical knowledge in our setting.

4 Methodology

4.1 Preliminaries: proving strategy

We adopt *natural deduction* (Prawitz, 1965) as a proof calculus. Here, a typical proving strategy is to decompose the logical formulas of T into atoms (subformulas with no logical connectives) and add them into a pool P of logical premises,

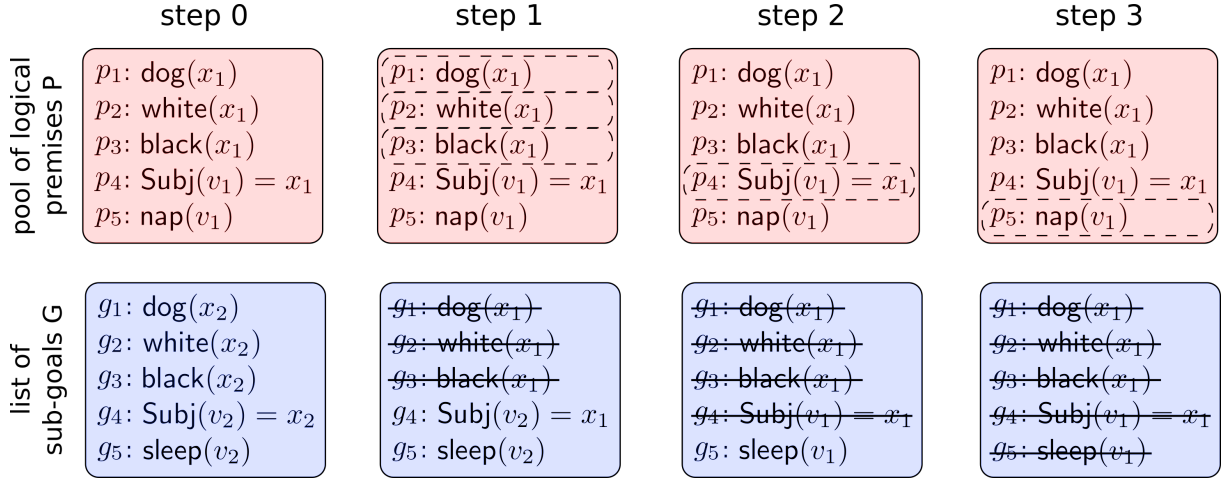


Figure 1: Trace of a proof in natural deduction. In step 0, T and H are decomposed into a pool of logical premises P and a list of sub-goals G . In step 1, g_1 , g_2 and g_3 are proved using p_1 , p_2 and p_3 and the variable unification $x_2 := x_1$. In step 2, g_4 is proved with p_4 and variable unification $v_2 := v_1$. Finally, g_5 can be proved from p_4 and the external axiom $\forall v. \text{nap}(v) \rightarrow \text{sleep}(v)$, resulting in a proved theorem.

$P = \{p_0(\theta_0), \dots, p_n(\theta_n)\}$, where p_i are predicates (function names) and θ_i are lists of (possibly structured) arguments of predicates p_i . The logical formula of H is similarly decomposed and its atoms are added either to the pool P or to a list of sub-goals $G = \{p'_0(\theta'_0), \dots, p'_m(\theta'_m)\}$.

As a running example, consider the T - H pair in (8) and (9), analysed as in (10) and (11):

- (8) A black and white dog naps.
(9) A black and white dog sleeps.
(10) $\exists x_1 v_1 (\text{dog}(x_1) \wedge \text{white}(x_1) \wedge \text{black}(x_1) \wedge \text{nap}(v_1) \wedge \text{Subj}(v_1) = x_1)$
(11) $\exists x_2 v_2 (\text{dog}(x_2) \wedge \text{white}(x_2) \wedge \text{black}(x_2) \wedge \text{sleep}(v_2) \wedge \text{Subj}(v_2) = x_2)$

As we can observe in Figure 1, T would be decomposed into the pool of logical premises

$$P = \{\text{dog}(x_1), \text{white}(x_1), \text{black}(x_1), \text{nap}(v_1), \text{Subj}(v_1) = x_1\}$$

and H into the list of sub-goals

$$G = \{\text{dog}(x_2), \text{white}(x_2), \text{black}(x_2), \text{sleep}(v_2), \text{Subj}(v_2) = x_2\}.$$

In general, existentially quantified formulas whose subformulas are connected only with logical conjunctions (e.g. $\exists \theta. A(\theta) \wedge B(\theta)$) are decomposed into subformulas $A(\theta)$ and $B(\theta)$, and added to P or G if they originate from T and H , respectively. Universally quantified formulas with logical implications (e.g. $\forall \theta. A(\theta) \rightarrow B(\theta)$) are not decomposed if such constructions appear in T ; if they

appear in H , $B(\theta)$ is added as a sub-goal in G and $A(\theta)$ is added as a logical premise in P . Decomposing higher-order constructions is possible, but we do not treat it here.

The proving then proceeds by selecting a sub-goal $p'_j(\theta'_j)$, searching P for a logical premise $p_i(\theta_i)$ for which p'_j and p_i and their arguments θ'_j and θ_i are equal (or they unify). If such a logical premise is found, then the sub-goal is proved and removed from G . That is the case of the sub-goals g_1 to g_4 in steps 1 and 2 of Figure 1, where predicates match those of p_1 to p_4 and variables unify as $x_2 := x_1$ and $v_2 := v_1$. If all sub-goals are proved, then the theorem is proved and the entailment judgement can be produced.

4.2 Detecting candidate sub-goals

However, there are theorems for which not all sub-goals can be proved. These cases occur when the source text fragment T does not entail the hypothesis H , or when there is a sub-goal for which no premise predicate matches. That is the case of sub-goal $g_5 : \text{sleep}(v_1)$ in Figure 1, which does not match any logical premise p_i . Due to the symbolic nature of logic provers, two different predicates with entailing semantics (e.g. nap and sleep) are considered unrelated, unless stated otherwise. For that reason, such a semantic relation, if it exists, needs to be made explicit in our framework.

In our natural deduction system, this operation is modeled as an *on-line axiom injection*, where candidate sub-goals are detected at proof-

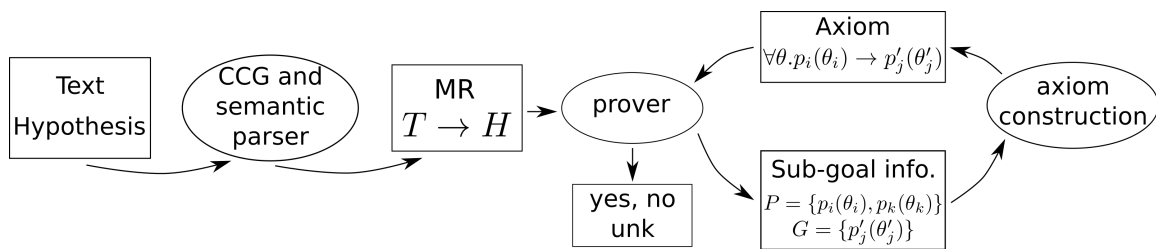


Figure 2: Pipeline for recognising textual entailment. Text and the Hypothesis are syntactically parsed with a CCG parser, and their logical meaning representations (MRs) are composed. A theorem $T \rightarrow H$ is constructed and a prover attempts to test it. If an unprovable sub-goal $p'_j(\theta'_j)$ is found, the axiom construction module attempts to build an axiom $\forall\theta.p_i(\theta_i) \rightarrow p'_j(\theta'_j)$ that is fed back into the theorem.

time, and their semantic relations (if any) with the premises are introduced in the form of axioms.

A sub-goal $p'_j(\theta'_j)$ is detected as a candidate to form an axiom if there is any logical premise $p_i(\theta_i)$ in P such that they share at least one argument, that is, $|\theta'_j \cap \theta_i| > 0$. Instead of requiring the set of arguments θ'_j and θ_i to be equal, we only require them to share at least one argument, to allow sub-goal predicates to underspecify arguments (e.g. drop the object or the subject of the sentence). The set R_j of possible relations between premise predicates p_i and a sub-goal predicate p'_j can then be defined as:

$$R_j = \{p_i \mid p_i(\theta_i) \in T \wedge |\theta'_j \cap \theta_i| > 0\} \quad (1)$$

In the example above, the sub-goal $\text{sleep}(v_1)$ is a candidate sub-goal to form an axiom, and its list of possible relations is $R_{\text{sleep}} = \{\text{nap}\}$.

4.3 On-demand axiom construction

Given a candidate sub-goal $p'_j(\theta'_j)$, R_j is a list of possible predicates that may semantically subsume or exclude the meaning of p'_j . At this point, we only need to classify each $p_i \in R_j$ as subsuming (entailing) p'_j , excluding (contradicting) it, or unrelated. In this work, we choose to use WordNet and VerbOcean (Chklovski and Pantel, 2004) as sources of external linguistic knowledge for their high precision. However, one could use other databases, ontologies or statistical classifiers, but we leave those considerations out of the scope of this paper.

There are two possible types of axioms that can be created: either entailing axioms $\forall\theta.p_i(\theta_i) \rightarrow p'_j(\theta'_j)$, or contradiction axioms $\forall\theta.p_i(\theta_i) \rightarrow \neg p'_j(\theta'_j)$, where $\theta = \theta'_j \cup \theta_i$ is the union of variable names occurring in θ'_j and θ_i . Entailing axioms are created when synonymy (e.g. $\text{house} \rightarrow \text{home}$),

hyponymy (e.g. $\text{sea} \rightarrow \text{water}$), adjectival similarity (e.g. $\text{huge} \rightarrow \text{big}$), derivationally related forms (e.g. $\text{accommodating} \rightarrow \text{accommodation}$), or inflection relations (e.g. $\text{wooded} \rightarrow \text{wood}$) are found in WordNet¹. Contradiction axioms are created solely when antonymy relations (e.g. $\text{big} \rightarrow \neg\text{small}$) are found. Once these axioms are created, they are inserted in the theorem and the proof continues.

Note that in Figure 1, if axioms were created a priori before the proof takes place, an axiom of the form $\forall x.\text{black}(x) \rightarrow \neg\text{white}(x)$ would have been created and a contradiction would be found in step 1 when proving the sub-goal $g_2 : \text{white}(x_1)$. We believe that the frequency of those cases increases with the length of sentences (or paragraphs) and the coverage of the external lexical resources.

Figure 2 shows our pipeline. Our software and Neo-Davidsonian semantic templates are open-sourced and publicly available at <https://github.com/mynlp/ccg2lambda>.

5 Experiments

5.1 Dataset

We use the SemEval-2014 version of the SICK dataset (Marelli et al., 2014), which is a dataset of English single-premise textual entailment problems annotated with three relations: *entailing* (yes), *contradicting* (no) or *unrelated* (unknown). The SICK dataset was originally developed to test approaches of compositional distributional semantics and it includes a variety of lexical, syntactic and semantic phenomena at the sentential level. With respect to FraCaS (Cooper et al., 1994), it contains less linguistically challenging problems but there is a higher need of lexical knowledge,

¹To maximise coverage, we consider all possible senses for a given predicate (word).

Problem ID	T-H pairs	Entailment
1412	T: <i>Men are sawing logs .</i> H: <i>Men are cutting wood .</i>	Yes
4114	T: <i>There is no man eating food .</i> H: <i>A man is eating a pizza .</i>	No
718	T: <i>A few men in a competition are running outside .</i> H: <i>A few men are running competitions outside .</i>	Unknown

Table 1: Examples of entailment problems from the SICK dataset. Some problems require a mix of logical reasoning and external lexical knowledge.

making it suitable to test our mechanism. With respect to the RTE datasets from the PASCAL RTE challenges, SICK problems are much shorter (and easier to syntactically parse), thus making them affordable for our current semantic parser.

Note that, although the SICK dataset only contains single-premise problems, our method also applies to multi-premise problems out-of-the-box. The dataset contains 4,500 problems for training, 500 for trial and 4,927 for testing, with a ratio of yes/no/unk problems of .29/.15/.56 in all splits. There are almost 212,000 running words, an average premise and conclusion length of 10.6 and a vocabulary of 2,409 words. Typically, there were about 3.6 words in the conclusion that did not appear in the premise, and 3.8 vice versa. Corpus statistics were collected after sentences were tokenized with the Penn Treebank Project tokenizer². Some examples of entailment problems for the SICK dataset are in Table 1.

5.2 Experimental setup

We parsed the tokenized sentences of the premises and hypotheses using the wide-coverage CCG parsers C&C (Clark and Curran, 2007) and EasyCCG (Lewis and Steedman, 2014). CCG derivation trees (parses) were converted into logical semantic representations using `ccg2lambda` (Martínez-Gómez et al., 2016) and our first-order Neo-Davidsonian event semantics. The validation of our version of semantic templates was carried out exclusively on the trial split of the SICK dataset.

We used Coq (Castéran and Bertot, 2004), an interactive natural deduction (Coquand and Huet, 1988) theorem prover that we run fully automatically with a number of built-in theorem-proving routines called *tactics*, which include first-order

²<https://www.cis.upenn.edu/~treebank/tokenization.html>

logic, arithmetic and equational reasoning. The axiom injection mechanism presented here could also have been implemented as a tactic to achieve a higher proving efficiency. However, this enhancement was left out from this work as it is both technically involved and makes our system bound to this specific prover. Instead, we monitor the proving progress and detect unprovable sub-goals; if our module produces an axiom, then it is introduced in the theorem and the proof is restarted. We call this method **SPSA**, the selector of predicates with shared arguments.

We use two in-house baselines: **No axioms** is our system without axiom injection, where only the logic of the language is used to prove sentence-level entailment relations. **Naïve** is a naïve method where we search for a WordNet linguistic relation between any two words of the premise and conclusion. If such a relation is found, then an axiom is constructed. All axioms found in this way are introduced in the theorem at once, and then the proving is performed. In this naïve method and in SPSA, if two words have more than one WordNet linguistic relation, then we only consider one, in this order: inflections, derivationally related forms, synonyms, antonyms, hypernyms, adjectival similarity and hyponyms. Moreover, although WordNet also contains linguistic relations between phrases, we only consider word-to-word relations. Our plain-logic system, the naïve and the SPSA methods were all timed-out after 100 seconds, at which the entailment judgement “unknown” was produced. When a syntactic parse error occurs, our systems tend to judge the entailment relation as “unknown”. To gain robustness and following Abzianidze (2015), we use a multi-parsing strategy (unless stated otherwise), that is, we use both C&C and EasyCCG parsers, and output any of their judgements if they are different

from “unknown”³.

Out of more than 20 participating teams in SemEval 2014, we compare our system to the following representative state-of-the-art systems: **Illinois-LH** (Lai and Hockenmaier, 2014), **ECNU** (Zhao et al., 2014), **UNAL-NLP** (Jiménez et al., 2014), **SemantiKLUE** (Proisl et al., 2014) are systems that build statistical classifiers on shallow features such as word alignments, syntactic structures and distributional similarities. These systems are the top performing systems in SemEval-2014. **The Meaning Factory** (Bjerva et al., 2014) is a hybrid system that combines logic semantic representations derived from CCG trees, with model builders and a statistical classifier, whereas **LangPro** (Abzianidze, 2015) is a purely logic system that composes Lambda Logical Forms of Natural Logic from CCG derivations. **Nutcracker** is a first-order logic system, where the effectiveness of introducing WordNet (and PPDB) using conventional methods is evaluated in (Pavlick et al., 2015).

We also include Markov Logic Networks (MLN) as described by Beltagy et al. (2016), where **MLN** denotes their system with closed-world assumptions and coreferences; **MLN-WN-PPDB** is their system augmented with WordNet and PPDB lexical relations, some hand-coded rules, and C&C/EasyCCG multi-parsing; **MLN-eclassif** denotes Beltagy et al. (2016)’s system augmented with a statistical classifier to recognise phrasal entailment relations (hence, we add this system in the list of statistical systems).

As it is common in RTE for SICK, we use precision and recall, where a successful prediction is one where the gold entailment label is either “yes” or “no”, and the system correctly predicts it. The accuracy, instead, is computed as a 3-way classification task, where a successful prediction counts on any of the three labels.

5.3 Results

Table 2 shows the results of our experimentation. Our plain first-order logic system **No axioms** has the highest precision 98.90%, but the lowest recall (46.48%). However, its accuracy (76.65%) is well beyond the baseline accuracy (56.69%) based on the majority class.

³If the system using C&C parser judges “yes” and the other judges “no”, or vice versa, then the final output is “unknown”.

System	Prec.	Rec.	Acc.
MLN-eclassif	–	–	85.10
Illinois-LH	81.56	81.87	84.57
ECNU	84.37	74.37	83.64
UNAL-NLP	81.99	76.80	83.05
SemantiKLUE	85.40	69.63	82.32
The Meaning Factory	93.63	60.64	81.60
LangPro Hybrid-800	97.95	58.11	81.35
MLN-WN-PPDB	–	–	80.40
Nutcracker-WN-PPDB	–	–	78.60
Nutcracker-WN	–	–	77.50
Nutcracker	–	–	74.30
MLN	–	–	73.40
Baseline (majority)	–	–	56.69
SPSA-VerbOcean	97.04	63.64	83.13
SPSA	97.07	62.13	82.97
SPSA, only C&C	97.27	58.48	81.44
SPSA, only EasyCCG	97.73	58.71	81.59
Naïve	92.99	59.70	80.98
No axioms	98.90	46.48	76.65

Table 2: Results on the test split of SICK dataset, using precision, recall and accuracy.

The **Naïve** method produced an increase of 4.33% points in accuracy with respect to the pure logic system. As a comparison, Pavlick et al. (2015) reported that a naïve introduction of axioms from WordNet on **Nutcracker** (Bjerva et al., 2014) for SICK dataset leads to an increase of 3.2% points of accuracy (from 74.3% to 77.5%), whereas using WordNet and sophisticated classifiers on the Paraphrase Database (Ganitkevitch et al., 2013) lead to an increase of 4.3% points in accuracy.

When the **SPSA** component substitutes the **Naïve** method, there is a 6.32% increase in the accuracy (from 76.63% to 82.97%), the recall increases by 15.65% and the precision only decreases by 1.83% with respect to the **No axioms** baseline. This system had higher performance than the other two best logic systems **The Meaning Factory** and **LangPro** (82.97% vs. 81.60% and 81.35%), and makes the use of external linguistic knowledge more effective than that in Pavlick et al. (2015), even without the use of a larger paraphrase database such as PPDB. If we add VerbOcean, which is an “unrefined” list of 22,306 verb relations, the accuracy further improves up to 83.13%, ranking our system on the fourth position among the statistical methods, af-

Prob. ID	T-H pairs	Gold	System	Axioms needed
1412	T: <i>Men are sawing logs</i> . H: <i>Men are cutting wood</i> .	Yes	Yes	$\forall v.\text{saw}(v) \rightarrow \text{cut}(v)$ $\forall x.\text{log}(x) \rightarrow \text{wood}(x)$
2404	T: <i>The lady is slicing a tomato</i> . H: <i>There is no one cutting a tomato</i> .	No	No	$\forall v.\text{slice}(v) \rightarrow \text{cut}(v)$
530	T: <i>A biker is wearing gear which is black</i> . H: <i>A biker wearing black is breaking the gears</i> .	Unk	Yes	
1495	T: <i>A man is playing a guitar</i> . H: <i>A man is strumming a guitar</i> .	Yes	Unk	$\forall v.\text{play}(v) \rightarrow \text{strum}(v)$
1266	T: <i>A band is playing on a stage</i> . H: <i>A band is playing onstage</i> .	Yes	Unk	“on a stage” \rightarrow “onstage”
2166	T: <i>A woman is sewing with a machine</i> . H: <i>A woman is using a machine made for sewing</i> .	Yes	Unk	“sewing with a machine” \rightarrow “using a machine made for sewing”
384	T: <i>A white and tan dog is running through the tall and green grass</i> . H: <i>A white and tan dog is running through a field</i> .	Yes	Unk	“tall and green grass” \rightarrow “field”

Table 3: Examples of successful and erroneous entailment predictions of our system, collected on the trial split of the SICK dataset.

ter **MLN-eclassif**, **Illinois-LH** and **ECNU**.

When limiting our semantic logical representations to those obtained only from the C&C parser (**SPSA, only C&C**), the recall was reduced by 3.65% and the accuracy by 1.53%, while the precision remained almost equal. Similar results were obtained with the EasyCCG parser. Although no single parser gives clearly a higher performance (in terms of recognising textual entailment), there are clear advantages to using both parsers, which is consistent with findings in Abzianidze (2015).

Regarding the proving time of the SPSA and the naïve methods, there were surprisingly no big differences. The proving time average, median and standard deviation per call to the theorem prover was 10 milliseconds, which was negligible when compared to the python overhead (the main language of our software). The SPSA method did an average of 10.7 calls to the theorem prover per RTE problem, whereas the naïve method did an average of 3.7. Note that these calls include the forward entailment and the contradiction proof attempts, both for C&C and EasyCCG parse trees. If lexical relations were found between the premise and conclusion, the naïve method would only do one more call (for each parser), whereas the SPSA method would do as many calls as axioms are potentially necessary. For that reason, the number of calls of the SPSA method is much larger.

5.4 Positive Examples and Error analysis

Table 3 shows some positive and negative examples of performance of our system on the trial split of the SICK dataset. For the first two examples, our plain logic system (without axioms)

produces incorrect entailment judgements (“unknown”), while our system produces the correct label, due to the introduction of two and one axioms in their corresponding theorems, respectively. The first axiom $\forall v.\text{saw}(v) \rightarrow \text{cut}(v)$ states that any event v of sawing is an event of cutting. Note that those predicates only have one argument event variable v , following Neo-Davidsonian event semantics. The second axiom $\forall x.\text{log}(x) \rightarrow \text{wood}(x)$ states that any entity x that is a log is wood.

In the third example, the label of the gold and plain logic system is “unknown”. However, our axiom injection system produces the axiom $\forall v.\text{wear}(v) \rightarrow \text{break}(v)$, where the meaning of wear is that of “impairment resulting from long use” (taken from WordNet). These two predicates apply over the object gear, thus sharing the same variable instantiation and producing the error. However, these cases are rare.

In the rest of the examples, our mechanism displays a lack of coverage to create axioms. In the fourth example, play and strum are not direct synonyms, but sister terms (according to WordNet). Many sister terms have an entailment relation, but many others do not (e.g. stand and run). In the rest of the examples, an ideal axiom injection mechanism would need access to string similarity methods (i.e. “on a stage” \rightarrow “onstage”) and to a knowledge base to understand that a machine is something that can be used, or that “tall and green grass” is a “field”.

6 Discussion and Future Work

The axiom injection method presented in this paper is more sophisticated and precise than simply assessing the linguistic relation between any two words from T and H and substituting those words conveniently (or equivalently in our framework, to introduce an axiom). Moreover, the naïve method is bound to show gradually lower precision when the size of sentences or the coverage of lexical resources increases, since there are more chances to obtain out-of-context lexical relations. Our axiom injection methods showed larger numbers of calls to the theorem prover, but each call took an average of only 10 milliseconds to complete. The reason for these larger numbers of calls reside in the implementation of the mechanism, since a proof needs to be re-run every time a new axiom is found. A possible enhancement would be to implement our axiom construction and injection as a Coq *tactic* that proves a sub-goal if its predicate has an entailing linguistic relation with any subset of the predicates in the logical premises at a specific stage of a proof.

In order to assess the precision of our SPSA method, we used WordNet and VerbOcean as our databases of external linguistic knowledge, which are databases of high-precision relations. In this setup, we found that our method solves effectively the lack of linguistic knowledge while keeping the precision high. However, other databases such as the Paraphrase Database (Ganitkevitch et al., 2013) or statistical classifiers could further increase the coverage of our method.

On one hand, the SPSA method requires access to the currently active sub-goals (and pool of logical premises) during a proof. Although such information is typically available in logic system, our method might not be directly applicable to systems that rely on statistical classifiers to judge compositional entailment relations. On the other hand, our system is characterised by its very high precision, which is a desirable characteristic when considering system combinations. In such setup, our system could run first, and if no conclusive sentential entailment relation is found, a statistical system could judge the relation, possibly using our logical representations and axioms as features.

Our method cannot be applied yet to larger texts, because CCG derivations accumulate errors when parsing larger sentences, and our logic composition is sensitive to those errors. Thus, making

our method more robust against CCG errors is a natural step. One possible solution is to use N-best CCG trees, collect features from those trees and possibly their semantic logical interpretations, and perform reranking.

7 Conclusion

We have presented a simple and effective method that introduces linguistic axioms on-demand to recognise textual entailments. The strategy is to build logical semantic representation of T and H , monitor the proof of the theorem $T \rightarrow H$, find unprovable sub-goals that share arguments (variable instantiations) with logical predicates, retrieve linguistic knowledge from an external resource, and insert the corresponding axioms on-demand. This system proved more effective and precise than simply enumerating all possible relations between words in T and H .

As it is common in logic systems, our method does not need parameter tuning. Moreover, the semantic representations and axioms are highly interpretable, which makes our system predictable, easy to understand, and easily extensible to use other linguistic resources or classifiers.

Finally, our logics and axiom construction/injection system have a high precision, making it a good candidate either as a standalone system, or as part of larger systems that use our logical semantic interpretations and axioms.

Acknowledgements

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and is also supported by CREST, JST. We thank the anonymous reviewers for their helpful comments.

References

- Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502, Lisbon, Portugal, September. Association for Computational Linguistics.
- Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. pages 11–21, June.

- Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrina Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42(4):763–808.
- Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI.
- Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635. Association for Computational Linguistics.
- Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 104–111. Association for Computational Linguistics.
- Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286.
- Pierre Castéran and Yves Bertot. 2004. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Springer Verlag.
- Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 33–40, Barcelona, Spain, July. Association for Computational Linguistics.
- Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.
- Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspers, Hans Kamp, Manfred Pinkal, Massimo Poesio, Stephen Pulman, et al. 1994. FraCaS—a framework for computational semantics. *Deliverable, D6*.
- Thierry Coquand and Gerard Huet. 1988. The calculus of constructions. *Information and Computation*, 76(2-3):95–120.
- Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing textual entailment: Models and applications*, volume 6. Morgan & Claypool Publishers.
- Yubing Dong, Ran Tian, and Yusuke Miyao. 2014. Encoding generalized quantifiers in dependency-based compositional semantics. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation*, pages 585–594.
- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.
- Sergio Jiménez, George Dueñas, Julia Baquero, and Alexander Gelbukh. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing*. Prentice-Hall, Inc.
- Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 990–1000, Doha, Qatar, October. Association for Computational Linguistics.
- Percy Liang, Michael Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.
- Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC2014*, pages 216–223.
- Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL-2016 System Demonstrations*, pages 85–90, Berlin, Germany, August. Association for Computational Linguistics.
- George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41, November.

- Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal, September. Association for Computational Linguistics.
- Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2016. Building compositional semantics and higher-order inference system for a wide-coverage Japanese CCG parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242, Austin, Texas, November. Association for Computational Linguistics.
- Terence Parsons. 1990. *Events in the Semantics of English*. MIT Press.
- Ellie Pavlick, Johannes Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. Adding semantics to data-driven paraphrasing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 1512–1522. Association for Computational Linguistics.
- Dag Prawitz. 1965. *Natural Deduction – A Proof-Theoretical Study*. Almqvist & Wiksell, Stockholm.
- Thomas Proisl, Stefan Evert, Paul Greiner, and Besim Kabashi. 2014. SemantiKLUE: Robust semantic similarity at multiple levels using maximum weight matching. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 532–540, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Mark Steedman. 2000. *The Syntactic Process*. MIT Press.
- Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 79–89, Baltimore, Maryland, June. Association for Computational Linguistics.
- Jiang Zhao, Man Lan, and Tiantian Zhu. 2014. ECNU: Expression- and message-level sentiment orientation classification in twitter using multiple effective features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 259–264, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Learning to Predict Denotational Probabilities For Modeling Entailment

Alice Lai and Julia Hockenmaier

Department of Computer Science
University of Illinois at Urbana-Champaign
{aylai2, juliahmr}@illinois.edu

Abstract

We propose a framework that captures the denotational probabilities of words and phrases by embedding them in a vector space, and present a method to induce such an embedding from a dataset of denotational probabilities. We show that our model successfully predicts denotational probabilities for unseen phrases, and that its predictions are useful for textual entailment datasets such as SICK and SNLI.

1 Introduction

In order to bridge the gap between vector-based distributional approaches to lexical semantics that are intended to capture which words occur in similar contexts, and logic-based approaches to compositional semantics that are intended to capture the truth conditions under which statements hold, Young et al. (2014) introduced the concept of “denotational similarity.” Denotational similarity is intended to measure the similarity of simple, declarative statements in terms of the similarity of their truth conditions.

From classical truth-conditional semantics, Young et al. borrowed the notion of the denotation of a declarative sentence s , $\llbracket s \rrbracket$, as the set of possible worlds in which the sentence is true. Young et al. apply this concept to the domain of image descriptions by defining the *visual denotation* of a sentence s as the set of images that s describes. The denotational probability of s , $P_{\llbracket s \rrbracket}(s)$, is the number of images in the visual denotation of s over the size of the corpus. Two sentences are denotationally similar if the sets of images (possible worlds) they describe have a large overlap. For example, “A woman is jogging on a beach” and “A woman is running on a sandy shore” can often be used to describe the same scenario, so they

will have a large image overlap that corresponds to high denotational similarity.

Given the above definitions, Young et al. estimate the denotational probabilities of phrases from FLICKR30K, a corpus of 30,000 images, each paired with five descriptive captions. Young et al. (2014) and Lai and Hockenmaier (2014) showed that these similarities are complementary to standard distributional similarities, and potentially more useful for semantic tasks that involve entailment. However, the systems presented in these papers were restricted to looking up the denotational similarities of frequent phrases in the training data. In this paper, we go beyond this prior work and define a model that can *predict* the denotational probabilities of novel phrases and sentences. Our experimental results indicate that these predicted denotational probabilities are useful for several textual entailment datasets.

2 Textual entailment in SICK and SNLI

The goal of textual entailment is to predict whether a hypothesis sentence is true, false, or neither based on the premise text (Dagan et al., 2013). Due in part to the Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2006), the task of textual entailment recognition has received a lot of attention in recent years. Although full entailment recognition systems typically require a complete NLP pipeline, including coreference resolution, etc., this paper considers a simplified variant of this task in which the premise and hypothesis are each a single sentence. This simplified task allows us to ignore the complexities that arise in longer texts, and instead focus on the purely semantic problem of how to represent the meaning of sentences. This version of the textual entailment task has been popularized by two datasets, the Sentences Involving Compositional Knowl-

edge (SICK) dataset (Marelli et al., 2014) and the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), both of which involve a 3-way classification for textual entailment.

SICK was created for SemEval 2014 based on image caption data and video descriptions. The premises and hypotheses are automatically generated from the original captions and so contain some unintentional systematic patterns. Most approaches to SICK involve hand-engineered features (Lai and Hockenmaier, 2014) or large collections of entailment rules (Beltagy et al., 2015).

SNLI is the largest textual entailment dataset by several orders of magnitude. It was created with the goal of training neural network models for textual entailment. The premises in SNLI are captions from the FLICKR30K corpus (Young et al., 2014). The hypotheses (entailed, contradictory, or neutral in relation to the premise) were solicited from workers on Mechanical Turk. Bowman et al. (2015) initially illustrated the effectiveness of LSTMs (Hochreiter and Schmidhuber, 1997) on SNLI, and recent approaches have focused on improvements in neural network architectures. These include sentence embedding models (Liu et al., 2016; Munkhdalai and Yu, 2017a), neural attention models (Rocktäschel et al., 2016; Parikh et al., 2016), and neural tree-based models (Munkhdalai and Yu, 2017b; Chen et al., 2016). In contrast, in this paper we focus on using a different input representation, and demonstrate its effectiveness when added to a standard neural network model for textual entailment. We demonstrate that the results of the LSTM model of Bowman et al. (2015) can be improved by adding a single feature based on our predicted denotational probabilities. We expect to see similar improvements when our predicted probabilities are added to more complex neural network entailment models, but we leave those experiments for future work.

3 Vector space representations

Several related works have explored different approaches to learning vector space representations that express entailment more directly. Kruszewski et al. (2015) learn a mapping from an existing distributional vector representation to a structured Boolean vector representation that expresses entailment as feature inclusion. They evaluate the resulting representation on lexical entailment tasks and on sentence entailment in SICK, but they re-

strict SICK to a binary task and their sentence vectors result from simple composition functions (e.g. addition) over their word representations. Henderson and Popa (2016) learn a mapping from an existing distributional vector representation to an entailment-based vector representation that expresses whether information is known or unknown. However, they only evaluate on lexical semantic tasks such as hyponymy detection.

Other approaches explore the idea that it may be more appropriate to represent a word as a region in space instead of a single point. Erk (2009) presents a word vector representation in which the hyponyms of a word are mapped to vectors that exist within the boundaries of that word vector’s region. Vilnis and McCallum (2015) use Gaussian functions to map a word to a density over a latent space. Both papers evaluate their models only on lexical relationships.

4 Denotational similarities

In contrast to traditional distributional similarities, Young et al. (2014) introduced the concept of “denotational similarities” to capture which expressions can be used to describe similar situations. Young et al. first define the *visual denotation* of a sentence (or phrase) s , $\llbracket s \rrbracket$, as the (sub)set of images that s can describe. They estimate the denotation of a phrase and the resulting similarities from FLICKR30K, a corpus of 30,000 images, each paired with five descriptive captions. In order to compute visual denotations from the corpus, they define a set of normalization and reduction rules (e.g. lemmatization, dropping modifiers, replacing nouns with their hypernyms, dropping PPs, extracting NPs) that augment the original FLICKR30K captions with a large number of shorter, more generic phrases that are each associated with a subset of the FLICKR30K images.

The result is a large subsumption hierarchy over phrases, which Young et al. call a denotation graph (see Figure 1). The structure of the denotation graph is similar to the idea of an entailment graph (Berant et al., 2012). Each node in the denotation graph corresponds to a phrase s , associated with its denotation $\llbracket s \rrbracket$, i.e. the set of images that correspond to the original captions from which this phrase could be derived. For example, the denotation of a phrase “*woman jog on beach*” is the set of images in the corpus that depict a woman jogging on a beach. Note that the deno-

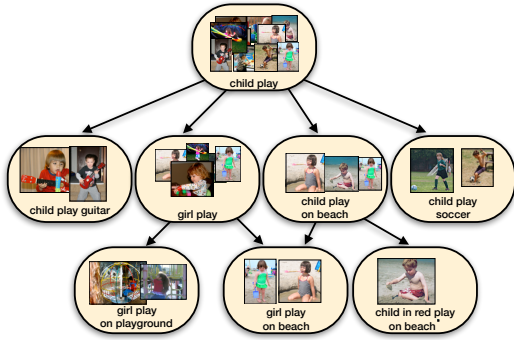


Figure 1: The denotation graph is a subsumption hierarchy over phrases associated with images.

tation of a node (e.g. “*woman jog on beach*”) is always a subset of the denotations of any of its ancestors (e.g. “*woman jog*”, “*person jog*”, “*jog on beach*”, or “*beach*”).

The denotational probability of a phrase s , $P_{\square}(s)$, is a Bernoulli random variable that corresponds to the probability that a randomly drawn image can be described by s . Given a denotation graph over N images, $P_{\square}(s) = \frac{\llbracket s \rrbracket}{N}$. The joint denotational probability of two phrases x and y , $P_{\square}(x, y) = \frac{\llbracket x \cap y \rrbracket}{N}$, indicates how likely it is that a situation can be described by both x and y . Young et al. propose to use pointwise mutual information scores (akin to traditional distributional similarities) and conditional probabilities $P_{\square}(x|y) = \frac{\llbracket x \cap y \rrbracket}{\llbracket y \rrbracket}$ as so-called denotational similarities. In this paper, we will work with denotational conditional probabilities, as they are intended to capture entailment-like relations that hold due to commonsense knowledge, hyponymy, etc. (what is the probability that x is true, given that y can be said about this situation?). In an ideal scenario, if the premise p entails the hypothesis h , then the conditional probability $P(h|p)$ is 1 (or close to 1). Conversely, if h contradicts p , then the conditional probability $P(h|p)$ is close to 0. We therefore stipulate that learning to predict the conditional probability of one phrase h given another phrase p would be helpful in predicting textual entailment. We also note that by the definition of the denotation graph, if x is an ancestor of y in the graph, then y entails x and $P_{\square}(x|y) = 1$.

Young et al. (2014) and Lai and Hockenmaier (2014) show that denotational probabilities can be at least as useful as traditional distributional similarities for tasks that require semantic inference such as entailment or textual similarity recognition. However, their systems can only use deno-

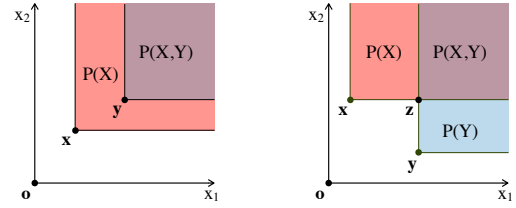


Figure 2: An embedding space that expresses the individual probability of events X and Y and the joint probability $P(X, Y)$.

tational probabilities between phrases that already exist in the denotation graph (i.e. phrases that can be derived from the original FLICKR30K captions).

Here, we present a model that learns to predict denotational probabilities $P_{\square}(x)$ and $P_{\square}(x|y)$ even for phrases it has not seen during training. Our model is inspired by Vendrov et al. (2016), who observed that a partial ordering \preceq over the vector representations of phrases can be used to express an entailment relationship. They induce a so-called order embedding for words and phrases such that the vector \mathbf{x} corresponding to phrase x is smaller than the vector \mathbf{y} , i.e. $\mathbf{x} \preceq \mathbf{y}$, for phrases y that are entailed by x , where \preceq corresponds to the reversed product order on \mathbb{R}_+^N ($\mathbf{x} \preceq \mathbf{y} \Leftrightarrow x_i \geq y_i \forall i$). They use their model to predict entailment labels between pairs of sentences, but it is only capable of making a binary entailment decision.

5 An order embedding for probabilities

We generalize this idea to learn an embedding space that expresses not only the binary relation that phrase x is entailed by phrase y , but also the probability that phrase x is true given phrase y . Specifically, we learn a mapping from a phrase x to an N -dimensional vector $\mathbf{x} \in \mathbb{R}_+^N$ such that the vector $\mathbf{x} = (x_1, \dots, x_N)$ defines the denotational probability of x as $P_{\square}(x) = \exp(-\sum_i x_i)$. The origin (the zero vector) therefore has probability $\exp(0) = 1$. Any other vector \mathbf{x} that does not lie on the origin (i.e. $\exists_i x_i > 0$) has probability less than 1, and a vector \mathbf{x} that is farther from the origin than a vector \mathbf{y} represents a phrase x that has a smaller denotational probability than phrase y . We can visualize this as each phrase vector occupying a region in the embedding space that is proportional to the denotational probability of the phrase. Figure 2 illustrates this in two dimensions. The zero vector at the origin has a probability pro-

portional to the entire region of the positive orthant, while other points in the space correspond to smaller regions and thus probabilities less than 1.

The joint probability $P_{\square}(x, y)$ in this embedding space should be proportional to the size of the intersection of the regions of \mathbf{x} and \mathbf{y} . Therefore, we define the joint probability of two phrases x and y to correspond to the vector \mathbf{z} that is the element-wise maximum of \mathbf{x} and \mathbf{y} : $z_i = \max(x_i, y_i)$. This allows us to compute the conditional probability $P_{\square}(x|y)$ as follows:

$$\begin{aligned} P_{\square}(x|y) &= \frac{P_{\square}(x, y)}{P_{\square}(y)} \\ &= \frac{\exp(-\sum_i z_i)}{\exp(-\sum_i y_i)} \\ &= \exp(\sum_i y_i - \sum_i z_i) \end{aligned}$$

Shortcomings We note that this embedding does not allow us to represent the negation of x as a vector. We also cannot represent two phrases that have completely disjoint denotations: in Figure 2, the $P(X)$ and $P(Y)$ regions will always intersect and therefore the $P(X, Y)$ region will always have an area greater than 0. In fact, in our embedding space, the joint probability represented by the vector \mathbf{z} will always be greater than or equal to the product of the probabilities represented by the vectors \mathbf{x} and \mathbf{y} . For any pair $\mathbf{x} = (x_1, \dots, x_N)$ and $\mathbf{y} = (y_1, \dots, y_N)$, $P_{\square}(X, Y) \geq P_{\square}(X)P_{\square}(Y)$:

$$\begin{aligned} P_{\square}(X, Y) &= \exp\left(-\sum_i \max(x_i, y_i)\right) \\ &\geq \exp\left(-\sum_i x_i - \sum_i y_i\right) \\ &= P_{\square}(X)P_{\square}(Y) \end{aligned}$$

(Equality holds when \mathbf{x} and \mathbf{y} are orthogonal, and thus $\sum_i x_i + \sum_i y_i = \sum_i \max(x_i, y_i)$). Therefore, the best we can do for disjoint phrases is learn an embedding that assumes the phrases are independent. In other words, we can map the disjoint phrases to two vectors whose computed joint probability is the product of the individual phrase probabilities.

Although our model cannot represent two events with completely disjoint denotations, we will see below that it is able to learn that some phrase pairs have very low denotational conditional probabilities. We note also that our model

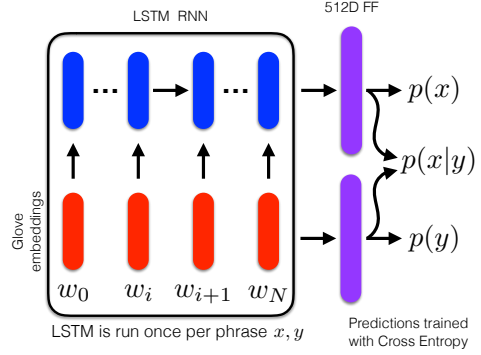


Figure 3: Our probability model architecture. Each phrase is a sequence of word embeddings that is passed through an LSTM to produce a 512d vector representation for the premise and the hypothesis. Both vectors are used to compute the predicted conditional probability and calculate the loss.

cannot express $P(X) = 0$ exactly, but can get arbitrarily close in order to represent the probability of a phrase that is extremely unlikely.

6 Our model for $P_{\square}(x)$ and $P_{\square}(x, y)$

We train a neural network model to predict $P_{\square}(x)$, $P_{\square}(y)$, and $P_{\square}(x|y)$ for phrases x and y . This model consists of an LSTM that outputs a 512d vector which is passed through an additional 512d layer. We use 300d GloVe vectors (Pennington et al., 2014) trained on 840B tokens as the word embedding input to the LSTM. We use the same model to represent both x and y regardless of which phrase is the premise or the hypothesis. Thus, we pass the sequence of word embeddings for phrase x through the model to get \mathbf{x} , and we do the same for phrase y to get \mathbf{y} . As previously described, we sum the elements of \mathbf{x} and \mathbf{y} to get the predicted denotational probabilities $P_{\square}(x)$ and $P_{\square}(y)$. From \mathbf{x} and \mathbf{y} , we find the joint vector \mathbf{z} , which we use to compute the predicted denotational conditional probability $P_{\square}(x|y)$ according to the equation in Section 5. Figure 3 illustrates the structure of our model.

Our training data consists of ordered phrase pairs $\langle x, y \rangle$. We train our model to predict the denotational probabilities of each phrase ($P_{\square}(x)$ and $P_{\square}(y)$) as well as the conditional probability $P_{\square}(x|y)$. Typically the pair $\langle y, x \rangle$ will also appear in the training data.

Our per-example loss is the sum of the cross entropy losses for $P_{\square}(x)$, $P_{\square}(y)$, and $P_{\square}(x|y)$:

$$L = -[P_{\square}(x) \log Q(x) + (1 - P_{\square}(x)) \log(1 - Q(x))] \\ - [P_{\square}(y) \log Q(y) + (1 - P_{\square}(y)) \log(1 - Q(y))] \\ - [P_{\square}(x|y) \log Q(x|y) + (1 - P_{\square}(x|y)) \log(1 - Q(x|y))]$$

We use the Adam optimizer with a learning rate of 0.001, and a dropout rate of 0.5. These parameters were tuned on the development data.

Numerical issues In Section 5, we described the probability vectors \mathbf{x} as being in the positive orthant. However, in our implementation, we use unnormalized log probabilities. This puts all of our vectors in the negative orthant instead, but it prevents the gradients from becoming too small during training. To ensure that the vectors are in \mathbb{R}_-^N , we clip the values of the elements of \mathbf{x} so that $x_i \leq 0$. To compute $\log P_{\square}(x)$, we sum the elements of \mathbf{x} and clip the sum to the range $(\log(10^{-10}), -0.0001)$ in order to avoid errors caused by passing $\log(0)$ values to the loss function. The conditional log probability is simply $\log P_{\square}(x|y) = \log P_{\square}(x, y) - \log P_{\square}(y)$, where $\log P_{\square}(x, y)$ is now the element-wise minimum:

$$\log P_{\square}(x, y) = \sum_i \min(x_i, y_i)$$

This element-wise minimum is a standard pooling operation (we take the minimum instead of the more common max pooling). Note that if $x_i > y_i$, neither element x_i nor y_i is updated with respect to the $P_{\square}(x|y)$ loss. Both x_i and y_i will always be updated with respect to the $P_{\square}(x)$ and $P_{\square}(y)$ components of the loss.

6.1 Training regime

To train our model, we use phrase pairs $\langle x, y \rangle$ from the denotation graph generated on the training split of the FLICKR30K corpus (Young et al., 2014). We consider all 271,062 phrases that occur with at least 10 images in the training split of the graph, to ensure that the phrases are frequent enough that their computed denotational probabilities are reliable. Since the FLICKR30K captions are lemmatized in order to construct the denotation graph, all the phrases in the dataset described in this section are lemmatized as well.

We include all phrase pairs where the two phrases have at least one image in common. These constitute 45 million phrase pairs $\langle x, y \rangle$ with $P_{\square}(x|y) > 0$. To train our model to predict

$P_{\square}(x|y) = 0$, we include phrase pairs $\langle x, y \rangle$ that have no images in common if $N \times P_{\square}(x)P_{\square}(y) \geq N^{-1}$ (N is the total number of images), meaning that x and y occur frequently enough that we would expect them to co-occur at least once in the data. This yields 2 million pairs where $P_{\square}(x|y) = 0$. For additional examples of $P_{\square}(x|y) = 1$, we include phrase pairs that have an ancestor-descendant relationship in the denotation graph. We include all ancestor-descendant pairs where each phrase occurs with at least 2 images, for an additional 3 million phrase pairs.

For evaluation purposes, we first assign 5% of the phrases to the development pool and 5% to the test pool. The actual test data then consists of all phrase pairs where at least one of the two phrases comes from the test pool. The resulting test data contains 10.6% unseen phrases by type and 51.2% unseen phrases by token. All phrase pairs in the test data contain at least one phrase that was unseen in the training or development data. The development data was created the same way.

This dataset is available to download at <http://nlp.cs.illinois.edu/HockenmaierGroup/data.html>.

We train our model on the training data (42 million phrase pairs) with batch size 512 for 10 epochs, and use the mean KL divergence on the conditional probabilities in the development data to select the best model. Since $P_{\square}(x|y)$ is a Bernoulli distribution, we compute the KL divergence for each phrase pair $\langle x, y \rangle$ as

$$D_{KL}(P||Q) = P_{\square}(x|y) \log \frac{P_{\square}(x|y)}{Q(x|y)} \\ + (1 - P_{\square}(x|y)) \log \frac{1 - P_{\square}(x|y)}{1 - Q(x|y)}$$

where $Q(x|y)$ is the conditional probability predicted by our model.

7 Predicting denotational probabilities

7.1 Prediction on new phrase pairs

We evaluate our model using 1) the KL divergences $D_{KL}(P||Q)$ of the gold individual and conditional probabilities $P_{\square}(x)$ and $P_{\square}(x|y)$ against the corresponding predicted probabilities Q , and 2) the Pearson correlation r , which expresses the correlation between two variables (the per-item gold and predicted probabilities) as a value between -1 (total negative correlation) and

	$P(x)$		$P(x y)$	
	KL	r	KL	r
Training data	0.0003	0.998	0.017	0.974
Full test data	0.001	0.979	0.031	0.949
Unseen pairs	0.002	0.837	0.048	0.920
Unseen words	0.016	0.906	0.127	0.696

Table 1: Our model predicts the probability of unseen phrase pairs with high correlation to the gold probabilities.

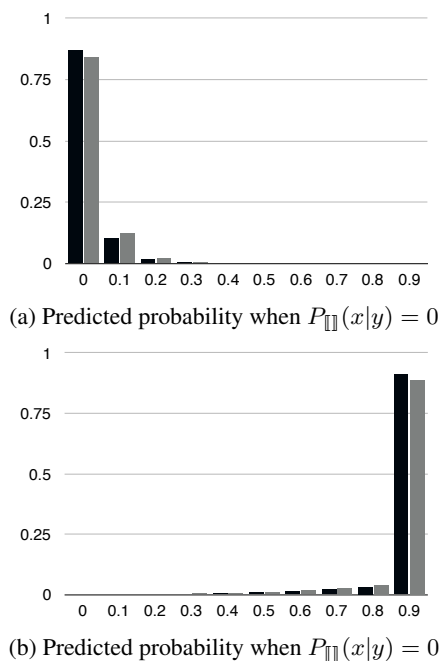


Figure 4: Predicted probabilities on denotational phrase test data when $P_{\text{DG}}(x|y) = 0$ is 0 or 1. Black is the full test data and gray is the subset of pairs where both phrases are unseen. Frequency is represented as a percentage of the size of the data.

1 (total positive correlation). As described above, we compute the KL divergence on a per-item basis, and report the mean over all items in the test set.

Table 1 shows the performance of our trained model on unseen test data. The full test data consists of 4.6 million phrase pairs, all of which contain at least one phrase that was not observed in either the training or development data. Our model does reasonably well at predicting these conditional probabilities, reaching a correlation of $r = 0.949$ with $P_{\text{DG}}(x|y)$ on the complete test data. On the subset of 123,000 test phrase pairs where both phrases are previously unseen, the model’s predictions are almost as good at $r = 0.920$.

On the subset of 3,100 test phrase pairs where at

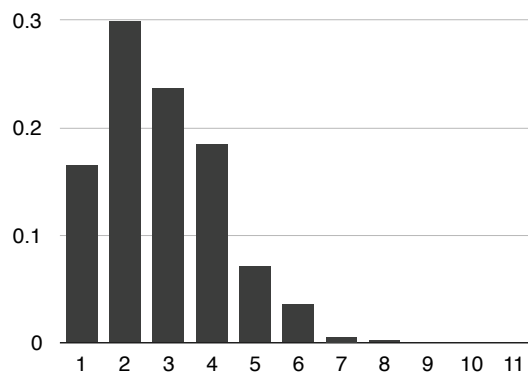


Figure 5: Distribution of phrase lengths as a fraction of the data size on the denotation graph phrase training data.

least one word was unseen in training, the model’s predictions are worse, predicting $P_{\text{DG}}(x|y)$ with a correlation of $r = 0.696$. On the remaining test pairs, the model predicts $P_{\text{DG}}(x|y)$ with a correlation of $r = 0.949$.

We also analyze our model’s accuracy on phrase pairs where the gold $P_{\text{DG}}(x|y)$ is either 0 or 1. The latter case reflects an important property of the denotation graph, since $P_{\text{DG}}(x|y) = 1$ when x is an ancestor of y . More generally, we can interpret $P_{\text{DG}}(h|p) = 1$ as a confident prediction of entailment, and $P_{\text{DG}}(h|p) = 0$ as a confident prediction of contradiction. Figure 4 shows the distribution of predicted conditional probabilities for phrase pairs where gold $P_{\text{DG}}(h|p) = 0$ (top) and gold $P_{\text{DG}}(h|p) = 1$ (bottom). Our model’s predictions on unseen phrase pairs (gray bars) are nearly as accurate as its predictions on the full test data (black bars).

7.2 Prediction on longer sentences

Our model up to this point has only been trained on short phrases, since conditional probabilities in the denotation graph are only reliable for phrases that occur with multiple images (see Figure 5 for the distribution of phrase lengths in the training data). To improve our model’s performance on longer sentences, we add the SNLI training data (which has a mean sentence length of 11 words) to our training data. We train a new model from scratch on a corpus consisting of the previously described 42 million phrase pairs and the 550,000 SNLI training sentence pairs (lemmatized to match our phrase pairs). We do not train on SICK because the corpus is much smaller and has a different distribution of phenomena, including

explicit negation. We augment the SNLI data with approximate gold denotational probabilities by assigning a probability $P_{\square}(S) = s/N$ to a sentence S that occurs s times in the N training sentences. We assign approximate gold conditional probabilities for each sentence pair $\langle p, h \rangle$ according to the entailment label: if p entails h , then $P(h|p) = 0.9$. If p contradicts h , then $P(h|p) = 0.001$. Otherwise, $P(h|p) = 0.5$.

Figure 6 shows the predicted probabilities on the SNLI test data when our model is trained on different distributions of data. The top row shows the predictions of our model when trained only on short phrases from the denotation graph. We observe that the median probabilities increase from contradiction to neutral to entailment, even though this model was only trained on short phrases with a limited vocabulary. Given the training data, we did not expect these probabilities to align cleanly with the entailment labels, but even so, there is already some information here to distinguish between entailment classes.

The bottom row shows that when our model is trained on both denotational phrases and SNLI sentence pairs with approximate conditional probabilities, its probability predictions for longer sentences improve. This model’s predicted conditional probabilities align much more closely with the entailment class labels. Entailing sentence pairs have high conditional probabilities (median 0.72), neutral sentence pairs have mid-range conditional probabilities (median 0.46), and contradictory sentence pairs have conditional probabilities approaching 0 (median 0.19).

8 Predicting textual entailment

In Section 7.2, we trained our probability model on both short phrase pairs for which we had gold probabilities and longer SNLI sentence pairs for which we estimated probabilities. We now evaluate the effectiveness of this model for textual entailment, and demonstrate that these predicted probabilities are informative features for predicting entailment on both SICK and SNLI.

Model We first train an LSTM similar to the 100d LSTM that achieved the best accuracy of the neural models in Bowman et al. (2015). It takes GloVe word vectors as input and produces 100d sentence vectors for the premise and hypothesis. The concatenated 200d sentence pair representation from the LSTM passes through three

Model	Test Acc.
Our LSTM	77.2
Our LSTM + CPR	78.2
Bowman et al. (2015) LSTM	77.2

Table 2: Entailment accuracy on SNLI (test).

Model	Test Acc.
Our LSTM	81.5
Our LSTM + CPR	82.7
Bowman et al. (2015) transfer	80.8

Table 3: Entailment accuracy on SICK (test).

200d tanh layers and a softmax layer for 3-class entailment classification. We train the LSTM on the SNLI training data with batch size 512 for 10 epochs. We use the Adam optimizer with a learning rate of 0.001 and a dropout rate of 0.85, and use the development data to select the best model.

Next, we take the output vector produced by the LSTM for each sentence pair and append our predicted $P_{\square}(h|p)$ value (the probability of the hypothesis given the premise). We train another classifier that passes this 201d vector through two tanh layers with a dropout rate of 0.5 and a final 3-class softmax classification layer. Holding the parameters of the LSTM fixed, we train this model for 10 epochs on the SNLI training data with batch size 512.

Results Table 2 contains our results on SNLI. Our baseline LSTM achieves the same 77.2% accuracy reported by Bowman et al. (2015), whereas a classifier that combines the output of this LSTM with only a single feature from the output of our probability model improves to 78.2% accuracy.

We use the same approach to evaluate the effectiveness of our predictions on SICK (Table 3). SICK does not have enough data to train an LSTM, so we combine the SICK and SNLI training data to train both the LSTM and the final model. When we add the predicted conditional probability as a single feature for each SICK sentence pair, performance increases from 81.5% to 82.7% accuracy. This approach outperforms the transfer learning approach of Bowman et al. (2015), which was also trained on both SICK and SNLI.

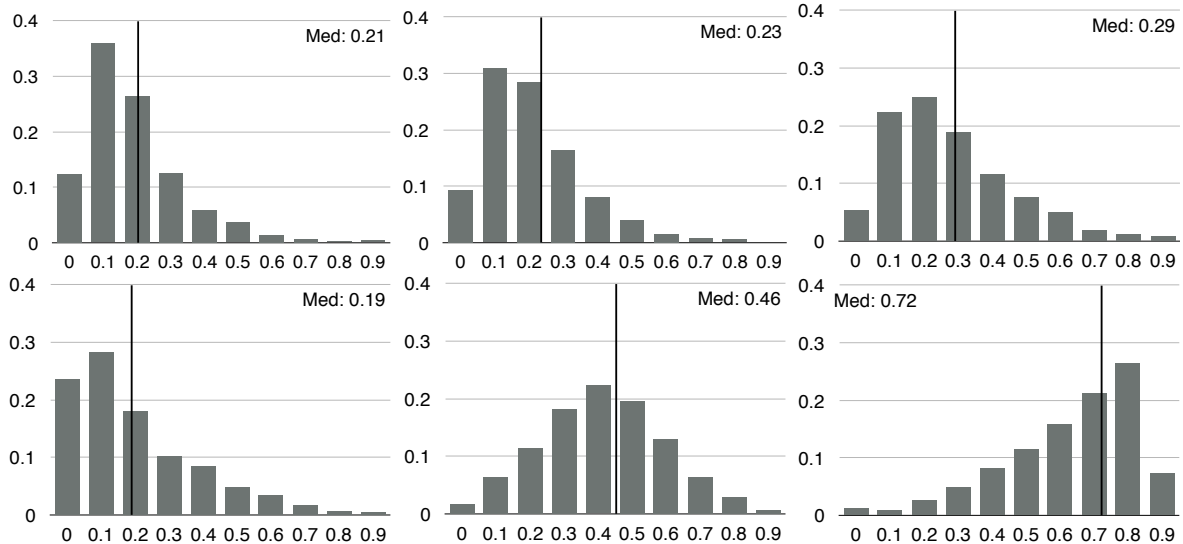


Figure 6: Predicted conditional probabilities $P(h|p)$ for SNLI sentence pairs (test) by entailment label, as a percentage of pairs with that label. Top: predictions from the model trained only on short denotational phrases. Bottom: predictions from the model trained on both short denotational phrases and SNLI.

Premise	Hypothesis	G	P
1 person walk on trail in woods	in forest	1.0	1.0
2 group of person bike	group of person ride	0.9	0.8
3 adult sing while play instrument	adult play guitar	0.8	0.8
4 person sit on bench outside	on park bench	0.4	0.4
5 tennis player hit ball	person swing	0.2	0.2
6 girl sleep	on pillow	0.1	0.2
7 man practice martial art	person kick person	0.1	0.3
8 person skateboard on ramp	man ride skateboard	0.2	0.2
9 busy intersection	city street	0.3	0.2
10 person dive into pool	person fly through air	0.1	0.1
11 sit at bench	adult read book	0.1	0.1
12 person leap into air	jump over obstacle	0.0	0.0
13 person talk on phone	man ride skateboard	0.0	0.0

Table 4: Gold and predicted conditional probabilities from the denotational phrase development data.

9 Discussion

Section 7 has demonstrated that we can successfully learn to predict denotational probabilities for phrases that we have not encountered during training and for longer sentences. Section 8 has illustrated the utility of these probabilities by showing that a single feature based on our model’s predicted conditional denotational probabilities improves the accuracy of an LSTM on SICK and SNLI by 1 percentage point or more. Although we were not able to evaluate the impact on more complex, recently proposed neural network models,

Premise	Hypothesis	Gold	Pred
skier on snowy hill	athlete	1.00	0.99
pitcher throw ball	mound	0.53	0.84
golf ball	athlete	0.53	0.66
person point	man point	0.48	0.41
in front of computer	person look	0.36	0.21

Table 5: Gold and predicted conditional probabilities from unseen pairs in the denotational phrase development data.

this improvement is quite encouraging. We note in particular that we only have accurate denotational probabilities for the short phrases from the denotation graph (mostly 6 words or fewer), which have a limited vocabulary compared to the full SNLI data (there are 5263 word types in the denotation graph training data, while the lemmatized SNLI training data has a vocabulary of 31,739 word types).

We examine examples of predicted conditional probabilities for phrase and sentence pairs to analyze our model’s strengths and weaknesses. Table 4 has example predictions from the denotation phrase development data. Our model correctly predicts high conditional probability for entailed phrase pairs even when there is no direct hypernym involved, as in example 2, and for closely related phrases that are not strictly entailing, as in example 3. Our model also predicts reasonable probabilities for events that frequently co-occur but are not required to do so, such as example 7.

	Premise	Hypothesis	CPR
Entailment	1 A person rides his bicycle in the sand beside the ocean.	A person is on a beach.	0.88
	2 Two women having drinks and smoking cigarettes at the bar.	Two women are at a bar.	0.86
	3 A senior is waiting at the window of a restaurant that serves sandwiches.	A person waits to be served his food.	0.61
	4 A man with a shopping cart is studying the shelves in a supermarket aisle.	There is a man inside a supermarket.	0.47
	5 The two farmers are working on a piece of John Deere equipment.	John Deere equipment is being worked on by two farmers.	0.16
Neutral	6 A group of young people with instruments are on stage.	People are playing music.	0.86
	7 Two doctors perform surgery on patient.	Two doctors are performing surgery on a man.	0.56
	8 Two young boys of opposing teams play football, while wearing full protection uniforms and helmets.	Boys scoring a touchdown.	0.30
	9 Two men on bicycles competing in a race.	Men are riding bicycles on the street.	0.24
Contradiction	10 Two women having drinks and smoking cigarettes at the bar.	Three women are at a bar.	0.79
	11 A man in a black shirt is playing a guitar.	The man is wearing a blue shirt.	0.47
	12 An Asian woman sitting outside an outdoor market stall.	A woman sitting in an indoor market.	0.22
	13 A white dog with long hair jumps to catch a red and green toy.	A white dog with long hair is swimming underwater.	0.09
	14 Two women are embracing while holding to go packages.	The men are fighting outside a deli.	0.06

Table 6: Predicted conditional probabilities for sentence pairs from the SNLI development data.

In examples 10 and 11, our model predicts low probabilities for occasionally co-occurring events, which are still more likely than the improbable co-occurrence in example 13. Table 5 demonstrates similar patterns for pairs where both phrases were unseen.

Table 6 has examples of predicted conditional probabilities for sentence pairs from the SNLI development data. Some cases of entailment are straightforward, so predicting high conditional probability is relatively easy. This is the case with example 2, which simply involves dropping words from the premise to reach the hypothesis. In other cases, our model correctly predicts high conditional probability for an entailed hypothesis that does not have such obvious word-to-word correspondence with the premise, such as example 1. Our model’s predictions are less accurate when the sentence structure differs substantially between premise and hypothesis, or when there are many unknown words, as in example 5. For neutral pairs, our model usually predicts mid-range probabilities, but there are some exceptions. In example 6, it is not certain that the people are playing music, but it is a reasonable assumption from the premise. It makes sense that in this case, our model assigns this hypothesis a higher conditional probability given the premise than for most neutral sentence pairs. In example 7, we might guess that the patient is a man with 50% probability, so the predicted conditional probability of our model seems reasonable. Our model cannot reason about numbers and quantities, as example

10 shows. It also fails to predict in example 11 that a man wearing a black shirt is probably not wearing a blue shirt as well. However, our model does correctly predict low probabilities for some contradictory examples that have reasonably high word overlap, as in example 13. Finally, example 14 shows that our model can correctly predict very low conditional probability for sentences that share no common subject matter.

10 Conclusion

We have presented a framework for representing denotational probabilities in a vector space, and demonstrated that we can successfully train a neural network model to predict these probabilities for new phrases. We have shown that when also trained on longer sentences with approximate probabilities, our model can learn reasonable representations for these longer sentences. We have also shown that our model’s predicted probabilities are useful for textual entailment, and provide additional gains in performance when added to existing competitive textual entailment classifiers. Future work will examine whether the embeddings our model learns can be used directly by these classifiers, and explore how to incorporate negation into our model.

Acknowledgments

This work was supported by NSF Grants 1563727, 1405883, and 1053856, and by a Google Research Award. Additional thanks to Yonatan Bisk and Pooya Khorrami.

References

- Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2015. Representing meaning with a combination of logical form and vectors. *CoRR*, abs/1505.06816.
- Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2012. Learning entailment relations by global graph structure optimization. *Computational Linguistics*, 38(1):73–111, March.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.
- Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree LSTM for natural language inference. *CoRR*, abs/1609.06038.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW’05, pages 177–190, Berlin, Heidelberg. Springer-Verlag.
- Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.
- Katrin Erk. 2009. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 57–65, Boulder, Colorado, June.
- James Henderson and Diana Popa. 2016. A vector space for distributional semantics for entailment. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2052–2062, Berlin, Germany, August.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.
- Germn Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.
- Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334.
- Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.
- Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.
- Tsendsuren Munkhdalai and Hong Yu. 2017a. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.
- Tsendsuren Munkhdalai and Hong Yu. 2017b. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.
- Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *The International Conference on Learning Representations (ICLR)*.
- Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *The International Conference on Learning Representations (ICLR)*.
- Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *The International Conference on Learning Representations (ICLR)*.
- Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 67–78.

A Societal Sentiment Analysis: Predicting the Values and Ethics of Individuals by Analysing Social Media Content

Tushar Maheshwari^{1*} Aishwarya N Reganti^{1*} Samiksha Gupta² Anupam Jamatia³
Upendra Kumar¹ Björn Gambäck^{4,5} Amitava Das¹

¹ Indian Institute of Information Technology, Sri City, Andhra Pradesh, India

² Indian Institute of Technology, Varanasi, Uttar Pradesh, India

³ National Institute of Technology, Agartala, Tripura, India

⁴ Norwegian University of Science and Technology, Trondheim, Norway

⁵ SICS Swedish ICT AB, Kista, Sweden

¹ {tushar.m14, aishwarya.r14, upendra.k14, amitava.das}@iiits.in

² samiksha.gupta.apm13@itbhu.ac.in

³ anupamjamatia@gmail.com

⁴ gamback@idi.ntnu.no

Abstract

To find out how users' social media behaviour and language are related to their ethical practices, the paper investigates applying Schwartz' psycholinguistic model of societal sentiment to social media text. The analysis is based on corpora collected from user essays as well as social media (Facebook and Twitter). Several experiments were carried out on the corpora to classify the ethical values of users, incorporating Linguistic Inquiry Word Count analysis, n-grams, topic models, psycholinguistic lexica, speech-acts, and non-linguistic information, while applying a range of machine learners (Support Vector Machines, Logistic Regression, and Random Forests) to identify the best linguistic and non-linguistic features for automatic classification of values and ethics.

1 Introduction

In the recent years, there have been significant efforts on determining the opinion/sentiment/emotion about a specific topic held by the author of a piece of text, and on automatic sentiment strength analysis of text, classifying it into either one of the classes positive, negative or neutral, or into Ekman's classes of happy, sad, anger, fear, surprise, and disgust. However, the intrinsic value of the lives we lead reflects the strength of our values and ethics which guide our social practices, attitude and behaviour. This paper reports work on investigating

a psycholinguistic model, the Schwartz model (Schwartz and Bilsky, 1990; Schwartz, 2012), and applying it to social media text. It will here be referred to as a societal sentiment model, since societal values grow from the interactions, and the views and sentiment of the society are key to ethical practices. No computational model for Schwartz' Values has been tested or examined before as such, but there has been a growing interest in the scientific community on doing automatic personality recognition, commonly using the Big 5 factor model (Goldberg, 1990) that defines personality traits such as openness, conscientiousness, extraversion, agreeableness, and neuroticism.

The Schwartz values model defines ten distinct ethical values (henceforth only values), that respectively are: *Achievement* sets goals and achieves them; *Benevolence* seeks to help others and provide general welfare; *Conformity* obeys clear rules, laws and structures; *Hedonism* seeks pleasure and enjoyment; *Power* controls and dominates others, controls resources; *Security* seeks health and safety; *Self-direction* wants to be free and independent; *Stimulation* seeks excitement and thrills; *Tradition* does things blindly because they are customary; *Universalism* seeks peace, social justice and tolerance for all.

Deeper understanding of human beliefs, attitudes, ethics, and values has been a key research agenda in Psychology and Social Science research for several decades. One of the most accepted and widely used frameworks is Schwartz 10-Values model, has seen great success in psychological research as well as in other fields. The ten basic values are related to various outcomes and ef-

* The two first authors contributed equally.

fects of a person's role in a society (Argandoña, 2003; Agle and Caldwell, 1999; Hofstede et al., 1991; Rokeach, 1973). Schwartz values have also proved to provide an important and powerful explanation of consumer behaviour and how they influence it (Kahle et al., 1986; Clawson and Vinson, 1978). Moreover, there are results that indicate how values of workforce and ethical practice in organisations are directly related to transformational and transactional leadership (Hood, 2003).

We believe that these kind of models may become extremely useful in the future for various purposes like Internet advertising (specifically social media advertising), community detection, computational psychology, recommendation systems, sociological analysis (for example East vs West cultural analysis) over social media.

In order to experiment with this, three corpora have been collected and annotated with Schwartz values. Two of the corpora come from popular social media platforms, Facebook and Twitter, while the third corpus consists of essays. A range of machine learning techniques has then been utilized to classify an individual's ethical practices into Schwartz' classes by analyzing the user's language usage and behaviour in social media. In addition to identifying the ten basic values, Schwartz' theory also explains how the values are interconnected and influence each other, since the pursuit of any of the values results in either an accordance with one another (e.g., Conformity and Security) or a conflict with at least one other value (e.g., Benevolence and Power). The borders between the motivators are artificial and one value flows into another. Such overlapping and fuzzy borders between values make the computational classification problem more challenging.

The paper is organized as follows. Section 2 introduces related work in the area. Details of the corpora collection and annotation are given in Section 3. Section 4 reports various experiments on automatic value detection, while Section 5 discusses the performance of the psycholinguistic experiments and mentions possible future directions.

2 Related Work

State-of-the-art sentiment analysis (SA) systems look at a fragment of text in isolation. However, in order to design a Schwartz model classifier, we require a psycholinguistic analysis. Therefore, textual features and techniques proposed and dis-

cussed for SA are quite different from our current research needs. Hence, we will here focus only on previous research efforts in automatic personality analysis that closely relate to our research work. Personality models can be seen as an augmentation to the basic definition of SA, where the aim is to understand sentiment/personality at person level rather than only at message level.

In recent years, there has been a lot of research on automated identification of various personality traits of an individual from their language usage and behaviour in social media. A milestone in this area was the 2013 Workshop and Shared Task on Computational Personality Recognition (Celli et al., 2013), repeated in 2014 (Celli et al., 2014). Two corpora were released for the 2013 task. One was a Facebook corpus, consisting of about 10,000 Facebook status updates of 250 users, plus their Facebook network properties, labelled with personality traits. The other corpus comprised 2,400 essays written by several participants labelled with the personalities. Eight teams participated in the shared task. The highest result was achieved by Markovikj et al. (2013) with an F-score of 0.73 (average for all the traits). The main methods and features (linguistic as well as non-linguistic) used by the participant groups were as follows.

Linguistic Features: The participating teams tested several linguistic features. Since n-grams are known to be useful for any kind of textual classification, all the teams tested various lengths of n-grams (uni, bi, and tri-grams). Categorical features like part-of-speech (POS), word level features like capital letters, repeated words were also used. Linguistic Inquiry Word Count (LIWC) features were used by all the teams as their baselines. LIWC (Pennebaker et al., 2015) is a hand-crafted lexicon specifically designed for psycholinguistic experiments. Another psycholinguistic lexicon called MRC (Wilson, 1988) was also used by a few teams, as well as lexica such as SentiWordNet (Baccianella et al., 2010) and WordNet Affect (Strapparava and Valitutti, 2004). Two more important textual features were discussed by the participating teams. Linguistic nuances, introduced by Tomlinson et al. (2013), is the depth of the verbs in the Wordnet troponymy hierarchy. Speech act features were utilized by Appling et al. (2013): the authors manually annotated the given Facebook corpus with speech acts and reported their correlation with the personality traits.

Here we briefly describe some people. Please read each description and think about how much each person is or is not like you. Tick the box to the right that shows how much the person in the description is like you.

	HOW MUCH LIKE YOU IS THIS PERSON?					
	Very much like me	Like me	Some- what like me	A little like me	Not like me	Not like me at all
1. Thinking up new ideas and being creative is important to her. She likes to do things in her original way. SD	6	5	4	3	2	1
2. It is important to her to be rich. She wants to have a lot of money and expensive things. PO	6	5	4	3	2	1
3. She thinks it is important that every person in the world be treated equally. She believes everyone should have equal opportunities in life. UN	6	5	4	3	2	1
4. Its important to her to show her abilities. She wants people to admire what she does. AC	6	5	4	3	2	1
5. It is important to her to live in secure surroundings. She avoids anything that might endanger her safety. SE	6	5	4	3	2	1

Table 1: Instructions and format of the Portrait Values Questionnaire. For each statement, the respondents should answer the question "How much like you is this person?" by checking one of the six boxes.

Non-Linguistic Features: All teams used Facebook network properties including network size, betweenness centrality, density and transitivity, provided as a part of the released dataset.

3 Corpus Acquisition

To start with, we ask a very fundamental question: *whether social media is a good proxy of the original (real life) society or not*. Back et al. (2010) and Golbeck et al. (2011) provide empirical answers to this question. Their results respectively indicate that, in general, people do not use virtual desired/bluffed social media profiles to promote an idealized-virtual-identity and that a user's personality can be predicted from his/her social media profile. This does not mean that there are no outliers, but our corpus collection was grounded on the assumption that it is true for a major portion of the population that social media behaviour to a large extent mirror that of the actual human society. Two of the most popular social media platforms, Twitter and Facebook, were chosen as sources for the corpora to validate this assumption. In addition, an essay corpus was collected. These three diverse corpora were then used for training and testing Schwartz values analysis methods.

3.1 Questionnaire for Self-Assessment

A standard method of psychological data collection is through self-assessment tests, popularly known as psychometric tests. In our experiments, self-assessments were obtained using male/female versions of PVQ, the Portrait Values Questionnaire (Schwartz et al., 2001). The participants

were asked to answer each question on a 1–6 Likert rating scale.¹ A rating of 1 means "not like me at all" and 6 means "very much like me". An example question is "He likes to take risks. He is always looking for adventures." where the user should answer while putting himself in the shoes of "He" in the question. A few exemplary items as well as the instructions and format of the written form of the PVQ are presented in Table 1.

The standard practice is to ask a fixed number of questions per psychological dimension. Here there are five questions for each of the ten Values classes, resulting in a 50 item questionnaire. Once all the questions in the PVQ have been answered, for each user and for each Values class, a score is generated by averaging all the scores (i.e., user responses) corresponding to the questions in that class, as described by Schwartz (2012). Further, the rescaling strategy proposed by Schwartz (2012) was used to eliminate randomness from each response given by a user as follows: For each user, the mean response score was first calculated considering all the responses s/he provided, and then the mean score from each response was subtracted. See Schwartz (2012) for more details on PVQ and the score computation mechanism.

The ranges of scores obtained from the previous rescaling method may vary across different Values classes. For instance, the ranges of the rescaled scores for the Essay corpus are as follows: Achievement [−4.12, 3.36], Benevolence [−1.56, 3.39], Conformity [−3.35, 3.01], Hedo-

¹<http://www.simplypsychology.org/likert-scale.html>

nism $[-5.18, 4.35]$, Power $[-6.0, 2.27]$, Security $[-2.60, 2.40]$, Self-Direction $[-1.61, 3.40]$, Stimulation $[-5.0, 2.63]$, Tradition $[-4.49, 3.35]$, and Universalism $[-3.33, 3.30]$.²

Hence the standard normalisation formula was applied to move the ranges of the different Values classes to the $[-1, 1]$ interval:

$$x_{scaled} = \frac{2 * (x - x_{min})}{x_{max} - x_{min}} - 1$$

A ‘Yes’ or ‘No’ binary value was assigned to each Values class: if the score was less than 0, the class was considered to be negative, indicating absence of that Values trait for the particular user; while scores ≥ 0 were considered to be positive, indicating the presence of that trait for the user. We will use the real scores ranging $[-1, 1]$ for the regression experiments mentioned in Section 4.

Reports of psychological analysis always depend on how the target population is chosen. Therefore while we are hypothesising that a few people are more Power oriented, an open question that remains unanswered is whom they are more Power oriented than. For example, if we (hypothetically) choose parliamentarians / politicians as participants in an experiment, then the entire examined population will likely turn out to be Power oriented. Therefore, it makes sense to normalise the obtained data into two groups $[-1, 0)$ and $[0, 1]$ and proclaim that people with $[0, 1]$ range scores are relatively more Power (or any other Value) oriented than the people having score ranging $[-1, 0)$.

The same normalisation mechanism was applied to all the corpora, but also after normalisation the different Values distributions were imbalanced (with the Facebook data being the most imbalanced). One possible reason behind such imbalanced distributions is that the portion of the real population using social media is slightly biased towards some Values types due to several societal reasons such as educational/family background, age group, occupation, etc. Another reason could be that the divisions between different value types simply never are balanced in any population. However, analysing such societal traits is a separate research direction altogether and out of the scope of the current study.

²The distribution of value types over a corpus was analysed using the Bienaymé-Chebyshev Inequality (Bienaymé, 1853; Tchébichef, 1867), showing that, e.g, most Achievement instances (89%) were in the range $[-2.96, 2.84]$.

The PVQ questionnaire setting described above was used to separately collect textual user data separately for the Essay, Facebook, and Twitter corpora, as discussed in the rest of this section.

3.2 Essay Corpus

The Essay corpus was collected using the Amazon Mechanical Turk (AMT)³ crowd-sourcing service. The turkers (users providing responses on AMT) were asked to compose an essay on the most important values and ethics guiding their lives, and to answer the PVQ questionnaire. A total of 981 users participated in the essay writing. However, not all the responses were useful for the analysis, since some participants did not answer all the questions and some did not write the essay carefully. For example, one user wrote: *“I don’t really have a guide in life. I go by what sounds and feels good. that means what makes me happy rather that effects others or not.”* Filtering out such users, data from 767 respondents was retained.

3.3 Twitter Corpus

In the first quarter of 2016, the micro blogging service Twitter averaged 310 million monthly active users,⁴ with around 6,000 tweets being posted every second. Therefore, Twitter came as the second natural choice as data source. The data collection was crowd-sourced using Amazon Mechanical Turk, while ensuring that the participants came from various cultures and ethnic backgrounds: the participants were equally distributed, and consisted of Americans (Caucasian, Latino, African-American), Indians (East, West, North, South), and a few East-Asians (Singaporeans, Malaysian, Japanese, Chinese). The selected Asians were checked to be mostly English speaking.

The participants were requested to answer the PVQ questionnaire and to provide their Twitter IDs, so that their tweets could be crawled. However, several challenges have to be addressed when working with Twitter, and a number of iterations, human interventions and personal communications were necessary. For example, several users had protected Twitter accounts, so that their tweets were not accessible when using the Twitter API. In addition, many users had to be discarded since they had published less than 100 tweets, making

³<https://www.mturk.com/>

⁴statista.com/statistics/282087/number-of-monthly-active-twitter-users

Corpus	AC	BE	CO	HE	PO	SE	SD	ST	TR	UN	Avg
Essay	65.70	52.41	63.10	54.40	40.40	52.50	54.00	52.50	43.20	54.43	66.10
Twitter	81.00	78.70	73.30	77.10	50.10	76.30	83.40	73.60	52.60	82.00	72.80
Facebook	88.13	93.22	91.52	86.44	46.67	98.30	89.83	86.44	71.18	94.91	84.67

Table 2: Flat distribution of Schwartz’ value types in the corpora: Achievement (AC), Benevolence (BE), Conformity (CO), Hedonism (H), Power (PO), Security (SE), Self-Direction (SD), Stimulation (ST), Tradition (TR), Universalism (UN). The last column gives the Average Majority Baselines.

Personality	AC	BE	CO	HE	PO	SE	SD	ST	TR	UN
Achievement (AC)	—	28.31	19.49	29.41	41.54	15.81	11.77	19.85	41.91	17.28
Benevolence (BE)	24.12	—	19.84	31.12	52.92	18.68	10.51	22.18	42.80	7.00
Conformity (CO)	18.59	23.42	—	35.32	47.58	12.64	17.47	24.91	35.32	15.99
Hedonism (HE)	17.60	24.04	25.32	—	43.35	21.03	9.01	14.60	45.92	12.88
Power (PO)	12.64	33.52	22.53	27.47	—	17.58	13.74	17.03	41.21	20.33
Security (SE)	17.63	24.82	15.47	33.81	46.04	—	13.31	21.94	38.49	14.39
Self-Direction (SD)	21.05	24.34	26.97	30.26	48.35	20.72	—	20.72	47.04	12.50
Stimulation (ST)	18.66	25.37	24.63	25.75	43.66	19.03	10.08	—	42.91	16.04
Tradition (TR)	18.13	23.83	9.84	34.72	44.56	11.40	16.58	20.73	—	17.10
Universalism (UN)	24.75	20.07	24.41	32.11	51.51	20.40	11.04	24.75	46.49	—

Table 3: Fuzzy distributions of Schwartz’ ten personality value types in the Twitter corpus.

them uninteresting for statistical analysis. In addition, some extreme cases when users mentioned someone else’s (some celebrity’s) Twitter account, had to be discarded. The open source free Twitter API: Twitter4J⁵ also has a limit of accessing only the current 3,200 tweets from any user. To resolve this issue, an open source Java application (Henrique, 2015) was used.

At the end of the data collection process, data from 367 unique users had been gathered. The highest number of tweets for one user was 15K, while the lowest number of tweets for a user was a mere 100; the average number of messages per user in the Twitter corpus was found to be 1,608.

3.4 Facebook Corpus

Facebook (FB) is the most popular social networking site in the world, with 1.65 billion monthly active users during the first quarter of 2016.⁶ Therefore, Facebook was a natural first choice for corpus collection, but since the privacy policy of Facebook is very stringent, accessing Facebook data is challenging. To collect the corpus, a Facebook Canvas web-application was developed using Facebook Graph API and Facebook SDK v5 for PHP library. Undergraduate students of two Indian institutes (NIT, Agartala, Tripura and IIIT, Sri City, Andhra Pradesh) were contacted for the data

collection. The application was circulated among the students and they were requested to take part in the PVQ survey and to donate their FB Timeline data and friend list data. Timeline data includes their own posts and all the posts they are tagged in, and posts other people made on their Timeline.

So far, data from 114 unique users has been collected, but the data is highly imbalanced (for some value types the distributions of ‘Yes’ and ‘No’ classes were in 90:10 ratio). Crowd-sourcing is a cheap and fast way to collect data, but unfortunately some annotators chose random labels to minimize their cognitive thinking load. These annotators can be considered as spammers and make aggregation of crowd-sourced data a challenging problem, as discussed in detail by Hovy et al. (2013). To filter out spammers, the MACE (Hovy et al., 2013) tool was used and data from 54 users discarded, so the final dataset includes only 60 participants. The average number of messages per user in the Facebook corpus is 681.

3.5 Corpus Statistics

Categorical flat distributions are reported in Table 2. Schwartz’ model defines fuzzy membership, which means that anyone having a Power orientation can have the Achievement orientation as well. To understand this notion vividly, we have reported the fuzzy membership statistics from the Twitter data in Table 3 (due to space limitations, statistics for the other corpora are not reported).

⁵<http://twitter4j.org/>

⁶<http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/>

Feature Ablation	AC	BE	CO	HE	PO	SE	SD	ST	TR	UN
Before Ablation	65.84	56.06	64.02	58.02	58.80	53.06	60.89	56.58	64.28	65.58
After Ablation	65.84	58.54	64.80	58.93	59.58	55.80	61.53	56.84	65.06	66.10
Number of features	52	37	65	38	54	47	65	53	39	48

Table 4: Best LIWC feature selection (by accuracy) for each of Schwartz’ ten personality value types. The values in the ‘Before Feature Ablation’ row are based on the full feature set (69 features).

The statistics in Table 3 show how the ten values are interconnected and influence each other, supporting the basic assumption of the Schwartz model that the borders between value classes are artificial and that one value flows into the next.

4 Automatic Identification Experiments

Several experiments were performed to get a better understanding of the most appropriate linguistic and non-linguistic features for the problem domain. The experiments were designed as a single label classification task (each input corresponds to one target label) with 20 classes, with ‘Yes’ and ‘No’ classes for each of the ten Schwartz values. Ten different classifiers were trained, each for a particular value type. Each classifier predicts whether the person concerned is positively or negatively inclined towards the given Schwartz value.

The versions implemented in WEKA (Witten and Frank, 2005) of three different machine learning algorithms were used in the experiments: Sequential Minimal Optimization (SMO; a version of Support Vector Machines, SVM), Simple Logistic Regression (LR), and Random Forests (RF). In all the mentioned experiments the corpora were pre-processed, i.e., tokenized by the CMU tokenizer (Gimpel et al., 2011) and stemmed by the Porter Stemmer (Porter, 1980). All the lexica were also stemmed in the same way before usage and all results reported below were obtained using 10-fold cross validation on each of the corpora.

4.1 Linguistic Features

LIWC Analysis: LIWC (Pennebaker et al., 2015) is a well developed hand-crafted lexicon. It has 69 different categories (emotions, psychology, affection, social processes, etc.) and almost 6,000 distinct words. The 69 categorical features were extracted as user-wise categorical word frequencies. As the text length (for the Essay corpus) or number of messages (Twitter and FB corpora) varies from person to person, Z-score normalization (or standardization) was applied using

the equation: $\hat{x} = (x - \mu)/\sigma$, where x is the ‘raw frequency count’, μ and σ are respectively the mean and standard deviation of a particular feature. After normalizing, each feature vector value is centered around $\sigma = (0, 1)$. This normalization led to an increase in the accuracy figures in many of the cases.

To investigate how each LIWC feature contributes, feature ablation was performed and the Pearson correlations of LIWC features vs value types were analysed. The final classifiers were trained using only the features that were contributing for a particular value type. This resulted in a performance boost and also gave reduced time complexity (both model training and testing times). Table 4 contains detailed categorical features for each value type for the SMO model, and, e.g., shows that the same accuracy (65.84%) for the Achievement class as obtained by using the full 69 feature set also can be obtained by using only 52 LIWC features. Moreover, the lowest obtained accuracy 53.06% for the Security class increased to 55.80% when considering only 47 features. Clearly, the details of which features actually contribute to each class cannot be included here (for space reasons), but the important lesson is that it is possible to reduce the feature set and increase performance in this way.

n-grams: In line with the systems discussed in Section 2, n-gram features were added to the LIWC baseline. In a first run, the top 20% of the most frequent uni-grams from the Essay corpus were included as new features, resulting in a 1452+69 feature set. Unexpectedly, SMO’s accuracy dropped by an average of 8.60%. The Achievement and Conformity values suffered the maximum performance drop, whereas Security and Hedonism had a slight increase in accuracy. Random Forests performed well in many of cases, except for the Security and Benevolence classes.

In a second iteration, categorical (value wise) n-grams features were selected and used. The resulting feature set sizes differ for each of the ten

values, ranging from the lowest number 886+69 for Power to the highest 1176+69 for Universalism. Marginally better performance was recorded.

n-grams (word grams) with various sizes of *n*, ranging from 2, 3, 4, 5, and so on, have different impact on performance on different kinds of applications. Commonly, bi-grams are better features for many text classification tasks. So, in a third iteration we tested system performance using bi-grams as added features with LIWC. As the total possible combinations of bi-grams are quite high, only the top 2,000 frequent bi-grams were included, resulting in 2000+69 features. There was no significant performance gain in this experiment on the Essay corpus, so this feature was not tested for the other two corpora.

Topic Modeling: In order to find out the bag-of-words features for each value type, i.e., the vocabulary that a person uses more frequently, the MALLET (McCallum, 2002)⁷ topic modelling toolkit was used to extract a number of topics. MALLET uses Gibbs Sampling and Latent Dirichlet Allocation (LDA). In a pre-processing stage, stop words were removed and case was preserved. For the Essay corpus, we tested with different number of topic clusters of sizes 10, 20, 50, 75, and 100, and observed that 50 was the most suitable number. Each of the 50 topics contained an average of 19 words (the topic key words identified by MALLET), each with a specific weight attached. The top 5 topics were chosen for each value type, according to these weights, and the words of these topics were added as a new feature set along with the LIWC baseline features.

It was also observed that the rankings of the top 5 topics were almost similar for each Schwartz value. The accuracies obtained were almost similar to the accuracies obtained in the previous experiments; however, this time, since the dimension of the feature set is much smaller, the time complexity decreased by almost a factor of 10. Hence the topic modelling was repeated for the social media corpora from Facebook and Twitter, but resulting in a different number of topic clusters, namely 89. Added to the 69 LIWC features this thus resulted in a total of 158 features.

Psycholinguistic Lexica: In addition to the base feature set from LIWC, two other psycholinguistic lexica were added: the Harvard General Inquirer

(Stone et al., 1966) and the MRC psycholinguistic database (Wilson, 1988). The Harvard General Inquirer lexicon contains 182 categories, including two large valence categories positive and negative; other psycholinguistic categories such as words of pleasure, pain, virtue and vice; words indicating overstatement and understatement, often reflecting presence or lack of emotional expressiveness, etc. 14 features from the MRC Psycholinguistic lexicon were included, namely, number of letters, phonemes and syllables; Kucera-Francis frequency, number of categories, and number of samples; Thorndike-Lorge frequency; Brown verbal frequency; ratings of Familiarity, Concreteness, Imagability and Age of acquisition; and meaningfulness measures using Colorado Norms and Pavio Norms. In order to get these MRC features a machine readable version of it has been used.⁸ Feature ranking was done by evaluating the contribution of each feature in an SMO classifier.

In addition, the sensorial lexicon Sensicon (Tekiroğlu et al., 2014) was used. It contains words with sense association scores for the five basic senses: Sight, Hearing, Taste, Smell, and Touch. For example, when the word 'apple' is uttered, the average human mind will visualize the appearance of an apple, stimulating the eye-sight, feel the smell and taste of the apple, making use of the nose and tongue as senses, respectively. Sensicon provides a numerical mapping which indicates the extent to which each of the five senses is used to perceive a word in the lexicon. Again, feature ablation was performed and the (Pearson) correlations of lexicon features vs values analysed. Finally, classifiers were trained using only contributing features for a particular value.

Speech Act Features: The way people communicate, whether it is verbally, visually, or via text, is indicative of Personality/Values traits. In social media, profile status updates are used by individuals to broadcast their mood and news to their peers. In doing so, individuals utilize various kinds of speech acts that, while primarily communicating their content, also leave traces of their values/ethical dimensions behind. Following the hypothesis of Appling et al. (2013), speech act features were applied in order to classify personalities/values. However, for this experiment the speech act classes were restricted to 11 major categories: Statement Non-Opinion (SNO), Wh

⁷<http://mallet.cs.umass.edu>

⁸<http://ota.oucs.ox.ac.uk/headers/1054.xml>

Speech Act	SNO	Wh	YN	SO	AD	YA	T	AP	RA	A	O	Avg.
Distribution	33.37	11.45	15.45	5.16	6.88	15.08	0.41	3.26	0.71	0.07	14.59	0.69
F ₁ -score	0.45	0.88	0.88	0.72	0.45	0.60	0.72	0.60	0.12	0.77	0.12	

Table 5: Speech act class distributions in the corpus (in %) and speech act classifier performance

Question (Wh), Yes-No Question (YN), Statement Opinion (SO), Action Directive (AD), Yes Answers (YA), Thanking (T), Appreciation (AP), Response Acknowledgement (RA), Apology (A) and others (O), hence avoiding having 43 fine-grained speech act classes.⁹

A corpus containing 7K utterances was collected from Facebook and Quora pages, and annotated manually. Motivated by the work by Li et al. (2014), this corpus was used to develop an SVM-based speech act classifier using the following features: bag-of-words (top 20% bigrams), presence of “wh” words, presence of question marks, occurrence of “thanks/thanking” words, POS tags distributions, and sentiment lexica such as the NRC lexicon (Mohammad et al., 2013), SentiWordNet (Baccianella et al., 2010), and WordNet Affect (Strapparava and Valitutti, 2004).

The categorical corpus distribution and the performance of the final classifier are reported in Table 5, showing an average F₁-score of 0.69 in 10-fold cross validation. Automatic speech act classification of social media conversations is a separate research problem, which is out of the scope of the current study. However, although the speech act classifier was not highly accurate in itself, the user specific speech act distributions (in %) could be used as features for the psycholinguistic classifiers (resulting in 11 additional features). Experiment on the Essay and Facebook corpora showed only 1.15% and 1% performance gain, respectively, whereas on the Twitter Corpus, a noticeable performance improvement of 6.12% (F-measure) was obtained. This indicates that speech acts are important signals of psychological behaviour, so even though the speech act classifier performs poorly, the extracted information is relevant.

4.2 Non-Linguistic Features

Social network structure is very useful to predict any person’s intrinsic value. For each user in the Twitter corpus, the total number of tweets or messages, total number of likes, average time differ-

ence between two tweets/messages, total number of favourites and re-tweets, and their in-degree and out-degree centrality scores on network of friends and followers were used as features adding to a total of 7 features along with the feature set used in the Topic Modelling experiment (69 LIWC + 89 Topic Modeling words from the Essay Corpus) after observation of the structure of tweets and the previously done linguistic feature experiments. The degree centrality was calculated as of a vertex v , for a given graph $G(V,E)$ with $|V|$ vertices and $|E|$ edges, is defined as: $\{C_D = deg(v)\}$.

The results of all the experiments after 10-fold cross-validation are summarized in Table 6 below.

5 Discussion and Conclusion

The main contributions of this paper are the introduction of a computational Schwartz values model, development of three different corpora annotated with Schwartz’ value, and experiments with features for automatic value classification. Table 6 reports that our models outperformed the majority baselines by significant margins of **+5.05**, **+7.20**, **+9.83** respectively on the Essay, Twitter and Facebook corpora. From the results it could be inferred that a few Schwartz values such as Self-Direction and Security are relatively difficult to identify, while on the other hand the accuracies for certain value types such as Power and Tradition are persistent and seem to be more salient.

The results also indicate that social media text is difficult for automatic classification, which is obvious from its terse nature. However, it is striking that the social media postings correlate far stronger than the essays with the psychometric data. This is probably since the size of the Twitter data is much larger than someones essay, and since when asked to write something, people become cautious; however, users behave more naturally when communicating in social media, making the data more insightful.

Another major implication from the experiments is that popular text classification features such as n-grams and topic-modelling were not performing well in this domain, indicating that this

⁹See for Fine-Gained Speech-Act classes <http://compprag.christopherpotts.net/swda.html>.

Values Classifier	Achievement			Benevolence			Conformity			Hedonism			Power			
	SMO	LR	RF	SMO	LR	RF	SMO	LR	RF	SMO	LR	RF	SMO	LR	RF	
LIWC	65.84	65.06	64.93	56.06	55.67	59.58	64.01	61.40	63.49	58.02	59.20	54.11	58.80	59.32	57.50	
+n-grams	57.50	62.71	65.84	55.54	53.19	58.80	56.45	61.54	64.80	58.28	58.41	58.02	53.46	59.71	58.41	
+Topic	58.54	64.15	65.32	54.37	53.46	59.06	60.63	62.32	63.75	58.80	58.41	58.28	58.15	57.76	56.71	
+Lexica	68.00	68.00	60.00	67.00	65.00	59.00	75.00	71.00	63.00	69.00	65.00	54.00	69.00	67.00	60.00	
+Speech-Act	68.00	66.80	60.30	69.00	67.00	59.00	71.00	67.00	59.00	68.00	67.00	60.00	70.00	67.00	58.00	
LIWC	TWT	80.93	80.93	80.10	78.75	78.75	77.38	73.02	72.48	77.93	77.11	76.84	76.02	54.77	50.68	52.59
	FB	85.60	82.90	81.60	89.10	88.20	89.90	87.50	86.60	87.50	85.70	80.20	80.20	67.40	59.20	59.30
+Topic	TWT	74.66	80.65	80.65	69.21	78.20	77.93	66.76	72.48	73.02	71.66	76.84	76.57	52.32	54.77	51.77
	FB	79.66	88.14	88.14	91.53	93.22	93.22	88.14	89.13	91.53	83.05	84.75	86.44	50.85	52.54	50.85
+Lexica	TWT	71.10	73.70	69.70	71.90	69.90	65.00	67.20	71.60	68.00	68.00	68.60	60.60	72.80	69.80	59.20
	FB	98.20	86.30	82.60	93.50	89.90	89.90	93.90	96.20	91.10	96.80	81.60	83.90	91.50	64.40	56.50
+Non-Linguistic	TWT	74.11	80.38	80.93	68.40	78.47	77.38	66.49	72.48	74.11	70.30	76.30	76.57	54.22	55.59	54.22
	FB	81.10	76.40	68.00	81.00	73.00	66.00	75.00	66.00	66.00	74.00	64.00	63.00	82.00	75.00	63.00
+Speech-Act	TWT	98.20	84.50	84.50	95.90	89.60	89.60	93.70	93.70	90.80	98.20	86.60	83.40	91.20	66.70	70.30
	FB	98.20	84.50	84.50	95.90	89.60	89.60	93.70	93.70	90.80	98.20	86.60	83.40	91.20	66.70	70.30
Values Classifier	Security			Self-Direction			Stimulation			Tradition			Universalism			Average
	SMO	LR	RF	SMO	LR	RF	SMO	LR	RF	SMO	LR	RF	SMO	LR	RF	
LIWC	53.06	55.02	56.06	60.89	59.84	58.54	56.58	56.98	56.45	64.28	65.97	64.02	65.58	65.71	65.32	61.36
+n-grams	56.84	56.45	56.71	56.06	58.54	58.41	56.06	56.67	56.71	58.67	65.06	64.28	58.28	65.45	65.84	61.05
+Topic	56.45	55.41	54.11	58.67	58.41	60.76	56.45	59.58	53.59	61.15	66.10	66.10	62.45	65.97	65.32	61.40
+Lexica	68.00	66.00	58.00	73.00	68.00	62.00	71.00	69.00	56.00	69.00	65.00	56.00	71.00	67.00	62.00	70.00
Speech-Act	73.00	69.00	58.00	69.00	66.00	55.00	75.00	71.00	63.00	74.00	70.00	62.00	72.80	68.30	61.50	71.15(+5.05)
LIWC	TWT	76.29	75.75	74.11	83.38	83.38	75.20	73.57	72.48	70.84	58.04	55.31	55.86	82.02	81.47	80.65
	FB	97.50	97.50	97.50	85.00	84.20	83.00	83.90	82.80	80.20	68.60	59.20	62.00	89.30	91.00	88.20
+Topic	TWT	70.57	74.93	75.48	76.84	83.38	83.38	64.12	72.47	71.66	52.04	53.95	59.67	74.93	81.47	81.20
	FB	93.22	98.30	98.30	86.44	84.75	89.83	81.36	84.75	86.44	62.71	74.58	71.19	89.83	94.91	93.22
+Lexica	TWT	70.60	74.30	69.50	75.60	74.40	76.60	68.80	68.60	68.30	73.90	69.50	62.30	78.00	82.20	76.30
	FB	97.50	97.50	97.50	91.60	82.40	85.00	92.80	83.90	83.90	84.60	75.10	78.90	90.70	92.40	91.60
+Non-Linguistic	TWT	71.18	74.66	75.20	76.57	83.38	83.38	65.58	73.57	71.66	52.59	53.41	55.86	74.39	81.74	82.02
	FB	78.00	80.00	69.00	78.00	76.00	75.00	73.00	66.00	68.00	80.00	71.00	63.00	89.00	81.10	77.00
+Speech-Act	TWT	97.90	97.40	97.40	93.90	83.60	84.50	96.30	85.20	83.94	91.10	71.30	78.20	89.50	91.30	92.20
	FB	97.90	97.40	97.40	93.90	83.60	84.50	96.30	85.20	83.94	91.10	71.30	78.20	89.50	91.30	92.20

Table 6: Automatic Schwartz value classification (accuracy) on the Essay, Twitter and Facebook corpora. Details of feature ablation and class wise performance.

is not yet another text classification problem, but that rather further deeper psycholinguistic analysis is required to find out hidden clues and the nature of language vs ethical practices. Here, it is worth noting the research by Pennebaker (2011), which indicates that, surprisingly, non-content words like pronoun, prepositions, particles, and even symbols are more salient indicators of our personality.

For the machine learners, closer analysis revealed that SMO’s performance was somehow irregular and random, which might be an indication of over-fitting. For example, the performance for some Schwartz values greatly decreased when adding n-grams as new features with LIWC, whereas some values showed the opposite behaviour, implying that each value type has its own set of distinct clues, but also high overlap. On the other hand, the performance of the Random Forests classifier increased when the number of features was increased, resulting in a larger forest and hence for most value types it performed better than the other two classifiers with less over-fitting.

A major limitation of the work is that the collected social network corpus is skewed. Reports of

psychological analysis on any community always depend on how the target population is chosen. It is absolutely impossible to get precisely balanced data from any real community. For example, it is rather impossible to have 150+ absolute power oriented people in a corpus of size 367 users data. The only solution to this problem is having more data, which we currently are collecting.

The data will be publicly released to the research community soon. We are also very keen on the applied side of this kind of models. Presently we are analysing the community detection problem in social media in relation to values. Another interesting application could be comparative societal analysis between the Eastern and Western regions of the world. Relations among personality and ethics could also be explored.

Acknowledgments

Thanks to the anonymous reviewers for their extensive and useful comments and suggestions regarding this article. We are also grateful for the support from all users who participated in the studies and volunteered their data.

References

- Bradley R. Agle and Craig B. Caldwell. 1999. Understanding research on values in business a level of analysis framework. *Business & Society*, 38(3):326–387.
- D. Scott Appling, Erica J. Briscoe, Heather Hayes, and Rudolph L. Mappus. 2013. Towards automated personality identification using speech acts. In *Seventh International AAAI Conference on Weblogs and Social Media*.
- Antonio Argandoña. 2003. Fostering values in organizations. *Journal of Business Ethics*, 45(1–2):15–28.
- Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*, Valletta, Malta, may. European Language Resources Association (ELRA).
- Mitja D. Back, Juliane M. Stopfer, Simine Vazire, Sam Gaddis, Stefan C. Schmukle, Boris Egloff, and Samuel D. Gosling. 2010. Facebook profiles reflect actual personality, not self-idealization. *Psychological Science*, 21:372–374.
- Irénée-Jules Bienaymé. 1853. *Considérations à l’appui de la découverte de Laplace sur la loi de probabilité dans la méthode des moindres carrés*. Imprimerie de Mallet-Bachelier.
- Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. 2013. The workshop on computational personality recognition 2013. In *Proceedings of the AAAI*, pages 2–5. AAAI.
- Fabio Celli, Bruno Lepri, Joan-Isaac Biel, Daniel Gatica-Perez, Giuseppe Riccardi, and Fabio Pianesi. 2014. The workshop on computational personality recognition 2014. In *Proceedings of the ACM International Conference on Multimedia*, pages 1245–1246. ACM.
- C. Joseph Clawson and Donald E. Vinson. 1978. Human values: A historical and interdisciplinary analysis. *Advances in Consumer Research*, 5(1).
- Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanagan, and Noah A Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.
- Jennifer Golbeck, Cristina Robles, and Karen Turner. 2011. Predicting personality with social media. In *CHI’11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’11, pages 253–262, New York, NY, USA. ACM.
- Lewis R. Goldberg. 1990. An alternative “description of personality”: the big-five factor structure. *Journal of personality and social psychology*, 59(6):1216.
- Jefferson Henrique. 2015. <https://github.com/Jefferson-Henrique/GetOldTweets-java>.
- Geert Hofstede, Gert Jan Hofstede, and Michael Minkov. 1991. *Cultures and organizations: Software of the mind*. McGraw-Hill, 2 edition.
- Jacqueline N. Hood. 2003. The relationship of leadership style and ceo values to ethical practices in organizations. *Journal of Business Ethics*, 43(4):263–273.
- Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL-HLT 2013*, pages 1120–1130.
- Lynn R. Kahle, Sharon E. Beatty, and Pamela Homer. 1986. Alternative measurement approaches to consumer values: The list of values (LOV) and values and life style (VALS). *Journal of consumer research*, pages 405–409.
- Jiwei Li, Alan Ritter, Claire Cardie, and Eduard Hovy. 2014. Major life event extraction from twitter based on congratulations/condolences speech acts. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1997–2007, Doha, Qatar, October. Association for Computational Linguistics.
- Dejan Markovikj, Sonja Gievska, Michal Kosinski, and David Stillwell. 2013. Mining Facebook data for predictive personality modeling. In *Proceedings of the 7th international AAAI conference on Weblogs and Social Media (ICWSM 2013)*, Boston, MA, USA.
- Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.
- James W. Pennebaker, Roger J. Booth, Ryan L. Boyd, and Martha E. Francis, 2015. *Linguistic Inquiry and Word Count: LIWC2015*. Pennebaker Conglomerates, Austin, Texas.
- James Pennebaker. 2011. *The Secret Life of Pronouns: What Our Words Say About Us*. Bloomsbury Publishing, New York, New York.
- Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

- Milton Rokeach. 1973. *The nature of human values*, volume 438. Free Press, New York.
- Shalom H. Schwartz and Wolfgang Bilsky. 1990. Toward a theory of the universal content and structure of values: Extensions and cross-cultural replications. *Journal of personality and social psychology*, 58(5):878.
- Shalom H. Schwartz, Gila Melech, Arielle Lehmann, Steven Burgess, Mari Harris, and Vicki Owens. 2001. Extending the cross-cultural validity of the theory of basic human values with a different method of measurement. *Journal of cross-cultural psychology*, 32(5):519–542.
- Shalom H. Schwartz. 2012. An overview of the Schwartz theory of basic values. *Online Readings in Psychology and Culture*, 2(1):11.
- Philip J. Stone, Dexter C. Dunphy, and Marshall S. Smith. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press.
- Carlo Strapparava and Alessandro Valitutti. 2004. WordNet Affect: an affective extension of WordNet. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal. European Language Resources Association (ELRA).
- Pafnuty Lvovich Tchébichef. 1867. Des valeurs moyennes (translated into French by N.V. Khanykov). *Journal de Mathématiques Pures et Appliquées*, 12(2):177–184.
- Serra Sinem Tekiroğlu, Gözde Özbal, and Carlo Strapparava. 2014. Sensicon: An automatically constructed sensorial lexicon. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1511–1521, Doha, Qatar, October. Association for Computational Linguistics.
- Marc T. Tomlinson, David Hinote, and David B. Bracewell. 2013. Predicting conscientiousness through semantic analysis of Facebook posts. In *Proceedings of the Workshop on Computational Personality Recognition*, pages 31–34.
- Michael Wilson. 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.
- Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Argument Strength is in the Eye of the Beholder: Audience Effects in Persuasion

Stephanie Lukin, Pranav Anand, Marilyn Walker and Steve Whittaker

Computer Science, Linguistics and Psychology Depts.

University of California, Santa Cruz

Santa Cruz, Ca. 95064

slukin, panand, mawalker, swhittak@ucsc.edu

Abstract

Americans spend about a third of their time online, with many participating in online conversations on social and political issues. We hypothesize that social media arguments on such issues may be more engaging and persuasive than traditional media summaries, and that particular types of people may be more or less convinced by particular styles of argument, e.g. emotional arguments may resonate with some personalities while factual arguments resonate with others. We report a set of experiments testing at large scale how audience variables interact with argument style to affect the persuasiveness of an argument, an under-researched topic within natural language processing. We show that belief change is affected by personality factors, with conscientious, open and agreeable people being more convinced by emotional arguments.

1 Introduction

Americans spend a third of their online time on social media, with many participating in online conversations about education, public policy, or other social and political issues. Our hypothesis is that online dialogs have important properties that may make them a useful resource for educating the public about such issues. For example, user-generated content might be more engaging and persuasive than traditional media, due to the prevalence of emotional language, social affiliation, conversational argument structure and audience involvement. Moreover, particular types of people may be more or less convinced by particular styles of argument, e.g. emotional arguments

may resonate with some personalities while factual arguments resonate with others.

Factual: Death Penalty	
Q1:	I'm sure there have been more repeat murderers than innocent people put to death. As far as the cost goes, is that really an issue? Execution Room = \$10,000. Stainless Steel Table = \$2,000. Leather Straps = \$200. Lethal Injection Chemicals = \$5,000. Knowing this person will never possibly be able to kill again = PRICELESS
R1:	Actually the room, straps, and table are all multi-use. And the drugs only cost Texas \$86.08 per execution as of 2002.
Emotional: Death Penalty	
Q2:	You mean, the perpetrator is convicted and the defender acquitted? Yes, that's the rule and not the exception. Notice here how no-one ended up dead, or even particularly seriously injured. Additionally the circumstances described are incredibly rare, that's why it makes the news.
R2:	The defender shouldn't even have been brought to trial in the first place. That doesn't make it any better. Somebody breaks into your home and threatens your family with rape and murder, they deserve serious injury at the very least.

Table 1: Factual vs. Emotional dialog exchanges 4forums.com. Q = Quote, R = Response.

For example, contrast the two informal dialogic exchanges about the death penalty in Table 1 with the traditional media professional summary in Table 2. We might expect the argument in Table 2 to be more convincing, because it is carefully written to be balanced and exhaustive (Reed and Rowe, 2004). On the other hand, it seems possible that people find dialogic arguments such as those in Table 1 more engaging and learn more from them. And indeed, about 90% of the people in online forums are so-called lurkers (Whittaker, 1996; Nonnecke and Preece, 2000; Preece et al., 2004), and do not post, suggesting that they are in fact **reading** opinionated dialogs such as those in Table 1 for interest or entertainment.

Research in social psychology identifies three

Curated Summary: Death Penalty	
PRO:	Proponents of the death penalty say it is an important tool for preserving law and order, deters crime, and costs less than life imprisonment. They argue that retribution or "an eye for an eye" honors the victim, helps console grieving families, and ensures that the perpetrators of heinous crimes never have an opportunity to cause future tragedy.
CON:	Opponents of capital punishment say it has no deterrent effect on crime, wrongly gives governments the power to take human life, and perpetuates social injustices by disproportionately targeting people of color (racist) and people who cannot afford good attorneys (classist). They say lifetime jail sentences are a more severe and less expensive punishment than death.

Table 2: Traditional balanced summary of the death penalty issue from ProCon.org.

factors that affect argument persuasiveness (Petty and Cacioppo, 1986; Petty and Cacioppo, 1988).

- the ARGUMENT itself
- the AUDIENCE
- the SOURCE of the argument

The ARGUMENT includes the content and its presentation, e.g. whether it is a monolog or a dialog, or whether it is factual or emotional as illustrated in Table 1 and Table 2. The AUDIENCE factor models people's prior beliefs and social affiliations as well as innate individual differences that affect their susceptibility to particular arguments or types of arguments (Anderson, 1971; Davies, 1998; Devine et al., 2000; Petty et al., 1981). Behavioral economics research shows that the cognitive style of the audience interacts with the argument's emotional appeal: emphasizing personal losses is more persuasive for neurotics, whereas gains are effective for extraverts (Carver et al., 2000; Mann et al., 2004). The SOURCE is the speaker, whose influence may depend on factors such as attractiveness, expertise, trustworthiness or group identification or homophily (Eagly and Chaiken, 1975; Kelman, 1961; Bender et al., 2011; Luchok and McCroskey, 1978; Ludford et al., 2004; McPherson et al., 2001).

We present experiments evaluating how properties of social media arguments interact with audience factors to affect belief change. We compare the effects of two aspects of the ARGUMENT: whether it is monologic or dialogic, and whether it is factual or emotional. We also examine how these factors interact with properties of the AUDIENCE. We profile audience **prior beliefs** to test if

more neutral people are swayed by different types of arguments than people with entrenched beliefs. We also profile the audience for Big Five personality traits to see whether **different personality types** are more open to different types of arguments, e.g., we hypothesize that people who are highly agreeable (A) might be more affected by the combative style of emotional arguments. We provide a new corpus for the research community of audience personality profiles, arguments, and belief change measurements.¹

Audience factors have been explored in social psychological work on persuasion, but have been neglected in computational work, which has largely drawn from sentiment, rhetorical, or argument structure models (Habernal and Gurevych, 2016b; Conrad et al., 2012; Boltuzic and Šnajder, 2014; Choi and Cardie, 2008). We demonstrate that, indeed, undecided people respond differently to arguments than entrenched people, and that the responses of undecided people correlate with personality. We show that this holds across an array of different arguments. Our research questions are:

- Can we mine social media to find arguments that change people's beliefs?
- Do different argument types have different effects on belief change?
- Do personality and prior beliefs affect belief change?
- Are different personality types differently affected by factual vs. emotional arguments?

Our results show a small but highly reliable effect that short arguments derived from online dialogs do lead people to change their minds about topics such as abortion, gun control, gay marriage, evolution, the death penalty and climate change. As expected, opinion change is greater for people who are initially more neutral about a topic, than those who are entrenched. However personality variables also have a clear effect on opinion change: neutral, balanced arguments are more successful with all personality types, but conscientious people are more convinced by dialogic emotional arguments, and agreeable people are more persuaded by dialogic factual arguments. We describe how we use plan these findings to select and repurpose social media arguments to adapt them to people's individual differences and thus maximize their educational impact.

¹nlds.so.e.ucsc.edu/persuasion_persona.

2 Related Work

Previous work on belief change has primarily focused on single, experimentally crafted, persuasive messages, rather than exploring whether user-generated dialogic arguments can be repurposed to persuade. Recently however several papers have begun to investigate two challenges in argument mining: (1) understanding the structure of an argument and extracting argument components (Lippi and Torroni, 2015; Nguyen and Litman, 2015; Stab and Gurevych, 2014; Lippi and Torroni, 2015; Biran and Rambow, 2011); and (2) understanding what predicts the persuasiveness of web-sourced argumentative content (Habernal and Gurevych, 2016b; Fang et al., 2016; Wachsmuth et al., 2016; Habernal and Gurevych, 2016a; Tan et al., 2016).

Tan et al. (2016) study belief change in the Reddit `/r/ChangeMyView` subreddit (CMV), in which an original poster (OP) challenges others to change his/her opinion. They build logistic regression models to predict argument success, identifying two conversational dynamic factors: a) early potential persuaders are more successful and b) after 4 exchanges, the chance of persuasion drops virtually to zero. Linguistic factors of persuasive posts include: a) dissimilar content words to the OP, b) similar stop words, c) being lengthy (in words, sentences, and paragraphs), d) italics and bullets. Finally, susceptibility to persuasion is correlated with singular vs. plural first person pronouns, which the authors relate to the personality trait of Openness to Experience. The CMV reddit offers a unique window into how persuasion of self-declared open-minded people occurs online. However, while Tan et al. find potential proxies for personality traits, they cannot examine traits directly because they do not have personality profiles as we do here. They also do not examine the effect of argument style as we do.

Recent work (Habernal and Gurevych, 2016b; Habernal and Gurevych, 2016a) also examines what makes an informal social media argument convincing. They have created a new dataset of pairs of arguments annotated for which argument is more convincing, along with the reasons given by annotators for its convincingness. They test several models for predicting convincingness comparing an SVM with engineered linguistic features to a BLSTM, with both models performing similarly. In contrast to our experiments, they do

not explore factors of the audience or explicitly vary the style of the argument.

Previous work also tests the hypothesis that dialogic exchanges might be more engaging, in the context of expository or car sales dialog (André et al., 2000; Lee, 2010; Craig et al., 2006; Stoyanchev and Piwek, 2010). Work comparing monologic vs. dialogic modes of providing information suggest that dialogs: (1) are more memorable and engaging, (2) stimulate the audience to formulate their own questions, and (3) allow audiences to be more successful at following communication (Lee et al., 1998; Fox Tree, 1999; Suzuki and Yamada, 2004; Driscoll et al., 2003; Fox Tree and Mayer, 2008; Fox Tree, 1999; Liu and Fox Tree, 2011).

Other work (Vydiswaran et al., 2012) explores how user-interface factors (e.g., number and order of argument presentation, whether and how arguments are rated) affect how readers process arguments. Several factors increased the number of passages read, including explicitly presenting contrasting viewpoints simultaneously. This exercise caused people with strong beliefs (about the healthiness of milk) to moderate their views after 20-30 minutes of concentrated study. We do not concentrate on interface factors, instead exploring how persuasiveness relates to audience factors and argumentative style. Also our experiments are run online with hundreds of users, rather than as a controlled study in the lab.

3 Experimental Method

Our experimental method consists of the following steps:

- Select user-generated dialogs with persuasive argument features from an online corpus of socio-political debates, exploring the role of **affect** (Sec. 3.1).
- Profile subjects for **personality traits** and **prior beliefs** about socio-political issues (Sec. 3.2).
- Expose subjects to user-generated, factual vs. emotional dialogic exchanges and compare the effects on belief change to balanced, curated arguments (Sec. 3.3).
- Conduct experiments to predict the degree of belief change as a function of prior belief, personality and type of argument.

The participants were pre-qualified using a reading comprehension task that checked their re-

sponses against a gold standard to ensure that they read the arguments carefully. Because we make many comparisons, and our experiments are conducted at large scale, all of our results incorporate Bonferroni corrections.

3.1 Dialog Selection: Identifying Socio-Emotional Arguments

Our work requires a new experimental corpus that is sensitive to readers' prior beliefs and personalities. We utilize online dialogs from `4forums.com` downloaded from The Internet Argument Corpus (IAC) (Walker et al., 2012c). The IAC contains quote/response pairs of targeted arguments between two people (Table 1) on topics such as: death penalty, gay marriage, climate change, abortion, evolution and gun control. Each argument is annotated to distinguish arguments making strong appeals to emotional factors versus straightforwardly factual arguments.

We selected a subset of extreme exemplars of factual (FACT) versus emotional (EMOT) arguments, defined as Q/R pairs reliably annotated to be at the extreme ends of the fact/emotion scale, i.e. responses with an average ≥ 4 annotation were considered factual, and those whose annotation averaged ≤ -4 were considered emotional on a scale of -5 to 5. Table 1 illustrates both factual (R1) and emotional (R2) arguments, with additional examples for other topics in Table 3.

In the IAC, 95% of the Q-R pairs are disagreements, the FACT and EMOT datasets were selected to contain a similar proportion. There was no correlation between agreement/disagreement and emotionality ($r = 0.07$, ns).

3.2 Personality

Personality is usually measured with a standardized survey that calculates a scalar value for the five OCEAN traits: openness to experience O, conscientiousness C, extraversion E, agreeableness A, and neuroticism N (Goldberg, 1990; Norman, 1963) We first conducted an experiment to profile the Big Five personality traits of 637 Turkers using the Ten Item Personality Inventory (TIPI) (Gosling et al., 2003). The TIPI instrument defines each person on a scale from 1 to 7 with 0.5 precision. In order to guarantee reliability of our results, we then verified that our pre-qualified Turkers are representative of the population as a whole, by comparing the means and standard deviations of our sample of 637 Turkers with the na-

Factual: Abortion	
Q3:	Not only that, to suggest that untold numbers of women would seek illegal abortions is a question-begging claim that has no grounding in history, logic, or reason. It is an unfounded, unproven claim. It is a betrayal to sound judgment to make decisions based upon unfounded predictions into the future.
R3:	But it is based on history. There is plenty of history showing that women had illegal abortions.
Factual: Climate Change	
Q4:	This is where the looney left gets lost. Their mantra is atmospheric CO2 levels are escalating and this is unquestionably causing earth's temperature rise. But ask yourself – if global temperatures are experiencing the biggest sustained drop in decades, while CO2 levels continue to rise – how can it be true?
R4:	Because internal variability from the likes of ENSO, which can cause short term swings of a full degree C, easily swamp the smaller increase we'd expect from CO2 forcing. Easy.
Emotional: Abortion	
Q5:	Undesired first pregnancy is an acute problem for many girls who choose to go under the surgical knife, even though that often ends up with infertility, broken life etc. Dry fasting is an alternative to first pregnancy abortion. If applied, up to 2-3 months old embryo gets dissolved after 15-16 days of the fast. Plus, there is no 'christian' sin.
R5:	No Christian sin??? Other then the intent to kill and then doing so:p
Emotional: Gay Marriage	
Q6:	Did anyone else expect anything less? These evil fundie christianists can have affairs, 2, 3, 4, or even 5 marriages yet gay people are a threat to marriage by wanting to get married.
R6:	You hear that cry...allowing gays to marry will cause the downfall of civilization...but you never hear 'how' or 'why'? More Chicken Little #####.

Table 3: Factual vs. Emotional dialog exchanges. Q = Quote, R = Response.

tional standards given in Gosling et. al (2003). Table 4 shows that our survey means and standard deviations are very close to the national norms, suggesting our sample is representative of the public in general, and hence can be used to validate whether social media arguments could fruitfully be used to educate the public.

	E	A	C	N	O
Our survey	4.30	5.19	5.50	4.82	5.53
Norms	4.44	5.23	5.4	4.83	5.38
Our survey σ	1.45	1.11	1.32	1.42	1.07
Norms σ	1.44	1.24	1.24	1.41	1.14

Table 4: TIPI σ and mean from our personality survey compared to the normal distribution

3.3 Prior Beliefs and Belief Change

Previous research suggests that people who are entrenched about an issue are unlikely to change their mind (Anderson, 1971; Davies, 1998; Devine et al., 2000), so we wanted to establish the baseline beliefs of our pre-qualified Turkers before they had been exposed to any arguments about a topic. We therefore collected each Turker’s initial stance on a topic, by asking them to answer a simple **stance question** with no context, for example: *Should the death penalty be allowed?*. Likert responses were recorded on a -5 to 5 slider scale with 0.01 degrees of precision, with labels on the slider of “Yes”, “No”, or “Neutral”.

Curated Summary: Abortion	
PRO:	Proponents, identifying themselves as pro-choice, contend that abortion is a right that should not be limited by governmental or religious authority, and which outweighs any right claimed for an embryo or fetus. They argue that pregnant women will resort to unsafe illegal abortions if there is no legal option.
CON:	Opponents, identifying themselves as pro-life, assert that personhood begins at conception, and therefore abortion is the immoral killing of an innocent human being. They say abortion inflicts suffering on the unborn child, and that it is unfair to allow abortion when couples who cannot biologically conceive are waiting to adopt.

Table 5: Traditional balanced summary of “Should abortion be legal?” from ProCon.org.

Our goal is to compare the belief change that results from social-media dialogs with the belief change from professionally-curated monologs. We selected the balanced, monologic, argument summaries from the website ProCon.org (in Table 2 with an additional example in Table 5). The arguments from ProCon.org are very high quality, and produced by domain experts.

After probing initial beliefs, we presented participants with one of the three different argument types to test their affect on belief change: a Curated Monolog (MONO) (Table 2), an emotional argument (EMOT) (R2 in Table 1), or a factual argument (FACT) (R1 in Table 1). After each person read one of these three types of arguments, we retested their reactions to the original stance question, while viewing the argument. Responses were again recorded on a -5 to 5 slider scale with 0.01 degrees of precision, with labels on the slider of “Yes”, “No”, or “Neutral”. We computed belief change by measuring differences in stance be-

	N	Mean change	σ change
MONO entrenched	1826	0.50	1.09
MONO neutral	1359	0.62	0.71
FACT entrenched	258	0.27	0.79
FACT neutral	202	0.39	0.55
EMOT entrenched	213	0.35	0.87
EMOT neutral	187	0.37	0.54
ALL entrenched	2951	0.43	1.00
ALL neutral	2234	0.51	0.65

Table 6: Means and σ for belief change for neutral and entrenched participants presented with MONO, FACT, or EMOT argument types. Neutrals show more belief change, and all argument types significantly affect beliefs

fore and after reading each argument. We created 20 HITs on Mechanical Turk for this task, with 5 items per hit.

4 Experimental Corpus Results

4.1 Entrenchment and Belief Change

Our first question is whether our method changed participant’s beliefs. Table 6 shows belief change as a function of argument type: monologs (MONO), factual (FACT) and emotional (EMOT). Belief change occurred for all argument types: and the change was statistically significant as measured by paired t-tests ($t_{(5184)} = 38.31$, $p < 0.0001$). This confirms our hypothesis that social media can be mined for persuasive materials. In addition, all three types of arguments independently led to significant changes in belief.²

One of the strongest theoretical predictions is that people with entrenched beliefs about an issue are less likely to change their mind when provided new information about that issue. Table 6 shows the relationship between initial beliefs and extent of belief change. We defined people as having more entrenched initial beliefs if their response to the initial stance question was within 0.5 points of the two ends of the scale, i.e. (1.0-1.5) or (4.5-5.0), indicating an extreme initial view.

We tested whether people who were more entrenched initially showed less change than those who were initially more neutral. We conducted a 2 Initial Belief (Entrenched/Neutral) X 3 Argument Type (MONO/EMOT/FACT) ANOVA, with Belief Change as the dependent variable, and Initial Belief and Argument Type as between subjects factors. Again, as expected, initially Entrenched

²(For MONO, $t_{(3184)} = 32.65$, $p < 0.0001$, for FACT, $t_{(1019)} = 14.81$, $p < 0.0001$, For EMOT, $t_{(979)} = 14.35$, $p < 0.0001$).

people showed less change ($M = 0.43$) than those who began with Neutral views ($M = 0.51$), ANOVA ($F_{(1,5179)}=5.97$, $p = 0.015$).

4.2 Argument Type and Belief Change

We wanted to test whether the engaging, socially interesting, dialogic materials of EMOT and FACT might promote more belief change than balanced curated monologic summaries. We tested the differences between argument types, finding a main effect for argument type ($F_{(2,5179)}=31.59$, $p < 0.0001$), with Tukey post-hoc tests showing MONO led to more belief change than both EMOT and FACT (both $p < 0.0001$), but no differences between EMOT and FACT overall across all subjects (See Table 6). Finally there was no interaction between Initial Belief and Argument Type ($F_{(2,5179)}=1.25$, $p > 0.05$): so although neutrals show more belief change overall, this susceptibility does not vary by argument type.

5 Predicting Belief Change

Our results so far show that our arguments changed people’s beliefs as a function of their prior beliefs and argument type. However we aim to automatically **predict** belief change, and hypothesize that knowing a person’s **personality** in combination with their **prior beliefs** will allow us to select social-media arguments that are more persuasive **for a particular individual**.

Thus, we vary whether providing a learner with features about a person’s personality improves performance for predicting belief change, when compared with providing information about degree of entrenchment alone. We use different representations for personality and prior beliefs as features, the raw score from the Likert slider for belief change and the TIPI score, as well as normalizations of the raw scores according to the distributions per topic, and finally categorical binning of the transformed scores.

5.1 Feature Development and Selection

New features were created by computing the z-transformation score from the raw prior beliefs and personality traits scores. Applying Equation 1 to the raw data creates a normal distribution where the new mean is 0 and the standard deviation is 1. For prior beliefs, x_i is an individual prior belief for a particular topic, \bar{x}_i is the mean, and σ_i is the standard deviation for the particular topic.

$$\frac{x_i - \bar{x}_i}{\sigma_{x_i}} \quad (1)$$

Categorical bins are derived from the transformed scores to describe the direction of the belief change by comparing prior and final recorded beliefs. The belief change is positive or negative depending upon where the Turkers rate themselves on the belief scale, moving more towards one side (1) or the other (5). Next, to control for variance, we apply a z-transformation on change scores to create a normal distribution. We classify the resulting distribution into three bins: Low, Medium, and High. The interpretation of what stance the Low and High bins represent is strictly topic dependent. The Medium bin consists of z-transformation values between -1 and 1. These are the people whose belief change is less than one standard deviation from the transformed mean. The Low bin contains z-transform scores of less than -1 and translates to belief changes of a large magnitude (more than a standard deviation from the mean) in a negative direction, where again, the meaning of “negative” is dependent upon how the question was framed. The High bin contains z-transformation scores of greater than 1 and translates to belief changes of a large magnitude in a positive direction. For example, the stance question *Should the death penalty be allowed?* has “no” at the -5 end and “yes” at the +5 end of the likert scale. A Low bin is indicative as moving in the direction of the “no” stance and High towards the “yes” stance.

Bins were also derived for the personality traits, e.g. for Openness, the High bin indicates someone who is very open, the Medium bin is average, and the Low bin is not open at all.

Finally, a binary feature was created to represent how entrenched an individual is in a particular topic. This feature is based on the raw prior belief score and is True if the prior belief score is within 0.5 points of either end point on the stance scale. This feature is different from the prior belief bins because this entrenchment feature groups together people who are in the extremes on both sides of the stance scale, while the prior belief bins distinguishes between the two ends.

We created a development set using data from a prior Mechanical Turk experiment which had 20 HITs, 5 questions per HIT, and 20 people who completed each HIT. In the same manner as the FACT and EMOT HITs, these Turkers (whose per-

sonality was already profiled) were asked about their prior beliefs about a topic, then presented with a factual or emotional argument. But in this case they were asked to rate the strength of the argument rather than to report their belief about the topic. We then identified the combination of features that best predicted argument strength in this development data, and then used this feature set for the belief change experiments below. Turkers who participated in this initial study did not participate in the belief change study and vice versa.

Results on the development set showed that the z-transformation scores for prior belief and personality performed better than the raw scores and bins. On the other hand, the belief change feature was most effective when represented as a categorical variable via binning and directionality. We also found that it is better to have **both** the z-transformed prior belief feature and the entrenchment feature. Thus our experiments below use these feature representations.

We test on three different datasets: MONO, FACT and EMOT, to elicit responses from reading the monologic summaries, and the factual and emotional dialogic arguments. The FACT and EMOT datasets have specific information in terms of scalar values about their degree of factuality or emotionality, on a scale of -5,+5 and a feature with this value is created for these datasets derived from the crowdsourced Turker judgments about the degree of Fact/Emotion in a Q/R pair, as described earlier. The monologic summaries (MONO) are assumed to be neutral and are not assigned a value for degree of factuality or emotionality.

5.2 Belief Change Experimental Results

Our dataset consists of 5185 items, with 3185 responses to the balanced MONO summaries, 1020 responses to FACT, and 980 responses to EMOT. We first applied 10-fold cross-validation with Naive Bayes, Nearest Neighbor, AdaBoost, and JRIP, from the Weka toolkit (Hall et al., 2005). Overall, Naive Bayes had the most consistent scores with our feature sets, thus we only report Naive Bayes experimental results below.

Seven feature sets were created for each of the three {MONO, FACT, EMOT} datasets. *None* feature sets are the no-personality baseline within each dataset. The baseline features contain **no information** about the personality of the unseen human subjects. We use the {MONO, FACT,

EMOT}+None feature sets for testing our hypothesis that personality affects belief change, and **our ability to predict belief change** using personality features. *All* feature sets have information about **all** of the human subjects' personality traits as 5 distinct features. The remaining five {O,C,E,A,N} feature sets examine the effect of providing information to the learner about personality using only **one personality trait at a time**, in order to determine if any personality trait is having a larger impact for belief change prediction.

Table 7 summarizes our key results, reporting accuracy, precision, recall, and F1 for predicting belief change as a discrete bin, Low, Medium, and High. We balanced each dataset to contain the same number of instances in bins, thus the accuracy for majority classification is 33% (Row 1).

After running Naive Bayes over all feature sets in the three datasets, we compared the experimental classifier performance of *All* and {O,C,E,A,N} against the None baselines using a Bonferroni corrected t-test for F1 measure. Using statistical ANOVA tests that control for pre and post test sample variance, we found small but highly reliable effects. We show all of our results, but focus our discussion below on statistically significance differences in F1. We boldface personality feature sets in Table 7 that are statistically significant when comparing {MONO, FACT, EMOT}+None with the other feature sets in the group.

The effect of argument alone (without personality information) can be seen by the no-personality baseline for each argument type, where we exclude personality information ({MONO, FACT, EMOT}+None). All these feature sets perform above the baseline of 33% (Row 1). This supports the results of our prior ANOVA testing over all subjects for belief change, and shows that the argument itself partially predicts belief change.

However, more interestingly, Table 7 also shows that providing the learner with information about personality consistently improves the ability of the learner to predict belief change. For all types of arguments, ie. the neutral, monologic summaries and the factual and emotional dialogs, the feature sets without any information about the personality traits of the unseen human subjects perform significantly worse than the feature sets that contain all five personality traits. MONO+None compared to MONO+All (rows 2 and 8 respectively) show a slight but significant increase in F1 from 0.51 to

row #	Dataset	TIPI	Accuracy	Precision	Recall	F1
1	Baseline		33%			
2	MONO	None	57%	0.50	0.58	0.51
3		Open	58%	0.52	0.59	0.52
4		Conscientious	58%	0.51	0.58	0.51
5		Extrovert	58%	0.49	0.57	0.49
6		Agreeable	58%	0.52	0.57	0.50
7		Neurotic	57%	0.49	0.55	0.47
8		All	58%	0.52	0.58	0.52
9		FACT	None	49%	0.47	0.47
10	Open		46%	0.45	0.47	0.46
11	Conscientious		48%	0.48	0.45	0.46
12	Extrovert		48%	0.46	0.45	0.44
13	Agreeable		51%	0.52	0.49	0.49
14	Neurotic		47%	0.45	0.44	0.43
15	All		50%	0.49	0.50	0.49
16	EMOT	None	53%	0.42	0.52	0.44
17		Open	56%	0.54	0.53	0.51
18		Conscientious	53%	0.49	0.51	0.48
19		Extrovert	49%	0.43	0.47	0.44
20		Agreeable	52%	0.48	0.50	0.48
21		Neurotic	53%	0.43	0.51	0.44
22		All	56%	0.55	0.57	0.56

Table 7: Predicting Belief Change with Naive Bayes for Three Data Sets: Statistics are based on 10-fold cross validation. Row numbers provided to reference particular results in the text.

0.52 (using a paired t-test on 10 fold cross validation scores, ($p = .001$)). Similarly, FACT+None versus FACT+All (rows 9 and 15) shows a significantly greater increase in F1: from 0.46 to 0.49 ($p = .0002$), as does EMOT+None versus EMOT+All (rows 16 and 22) with F1 increasing from 0.44 to 0.56 ($p = .00001$). This confirms that the personality traits improves a model’s ability to predict belief change in unseen human subjects.

Next we compared the effects of providing the learner with information about each individual personality feature **in isolation** by comparing {MONO, FACT, EMOT}+None with individual personality factors. For MONO, we found that adding personality information about Openness to Experience (MONO+O, row 3) improved F1 from 0.51 to 0.52 compared with a no-personality baseline ($p = .0006$). This suggests that open people are more persuaded by balanced monologic arguments.

A more interesting result is that Openness to Experience (EMOT+O, row 17) was also important for Emotional arguments, increasing F1 from 0.44 to 0.51 ($p = .00001$). In contrast, Openness had no effect for Factual arguments (Row 10) ($p > 0.05$). Models for predicting belief change for Emotional arguments also benefit from information about Conscientiousness and Agreeableness. Row 17 (EMOT+O), Row 18 (EMOT+C) and Row 20 (EMOT+A) all show significant differences in

F1, with EMOT+O better than EMOT+None ($p = .00001$), EMOT+C better than EMOT+None ($p = .00001$) and EMOT+A better than EMOT+None ($p = .0001$).

Information about Agreeableness also improves the quality of the belief change models for the factual dialogs (FACT+A, row 5) with an increase in F1 from 0.46 baseline to 0.49 ($p = .004$), suggesting that people who are more Agreeable are more influenced by factual arguments. This confirms one of our initial hypotheses that Agreeable people would be more sensitive to the fact/emotional dimension of arguments because of their desire to either avoid conflict (highly Agreeable people) or to seek conflict (Disagreeable people).

6 Conclusions

To the best of our knowledge we are the first to examine the interaction of social media argument types with audience factors. Our contributions are:

- A new corpus of personality information and belief change in socio-political arguments;
- A new method for identifying and deploying social media content to inform and engage the public about important social and political topics;
- Results showing at scale (hundreds of users) that we can mine arguments from online discussions to change people’s beliefs;

- Results showing that different types of arguments have different effects: while balanced monologic summaries led to the greatest belief change, socio-emotional online exchanges also caused changes in belief.

Although our short question/response pairs did not induce as much belief change as the curated balanced monologs, we believe that these are striking results given that the materials we extracted from online discussions are not balanced or professionally produced, but instead are simple fragments extracted from online discussions.

Further, confirming prior work on persuasion (Eagly and Chaiken, 1975; Kelman, 1961; Petty et al., 1981), we found that these effects depend on audience characteristics. As expected, belief depended on the strength of prior beliefs so that initially neutral people were more likely to be persuaded than entrenched individuals, regardless of the type of argument. Again supporting our predictions, argument effectiveness depended on personality type. People who are Open to Experience were influenced by balanced and emotional materials. In contrast, Agreeable people are most affected by factual materials. Emotional arguments had very different effects from factual and balanced monologs: Openness is important but so too are Conscientiousness and Agreeableness.

How can we explain this? People who are more Open are typically receptive to new ideas. But our results for emotional arguments also show that Conscientious people change their views when presented with emotional arguments, possibly because they are careful to process the arguments however expressed. And Agreeable people may also be motivated to change belief by emotional arguments because they are less likely to be influenced by personal feelings.

Our results have numerous implications that suggest further technical experimentation. The fact that we can induce belief change by extracting simple discussion fragments suggests that belief change can be induced without the application of sophisticated text processing tools. While our results for balanced monologs suggest that summaries increase belief change, summary tools for such arguments are still under development (Misra et al., 2015). However, perhaps high quality summaries may not be needed if compelling argument fragments can be automatically extracted (Misra et al., 2016b; Subba and Di Eugenio, 2007; Nguyen

and Litman, 2015; Swanson et al., 2015).

Our work also suggests the importance of personalization for persuasion: with different personality types being open to different styles of argument. Future work might be based on methods for profiling participant personality from simple online behaviors (Di Eugenio et al., 2013; Liu et al., 2016; Pan and Zhou, 2014; Yee et al., 2011), or from user-generated content such as first-person narratives or conversations (Mairesse and Walker, 2006a; Mairesse and Walker, 2006b; Rahimtoroghi et al., 2016; Rahimtoroghi et al., 2014). We could then select personalized arguments to meet a participant's processing style.

While here we used crowdsourced judgments to select arguments of particular types. Elsewhere, we present algorithms for automatically identifying and bootstrapping arguments with different properties. We have methods to extract arguments that represent different stances on an issue (Misra et al., 2016a; Anand et al., 2011; Sridhar et al., 2015; Walker et al., 2012a; Walker et al., 2012b), as well as argument exchanges that are agreements vs. disagreements (Misra and Walker, 2015), factual vs. emotional arguments (Oraby et al., 2015), sarcastic and not-sarcastic arguments, and nasty vs. nice arguments (Oraby et al., 2016; Lukin and Walker, 2013; Justo et al., 2014).

An open question is to whether these effects are long term. Our approach limits us to examining belief change during a single session for practical reasons; long-term cross-session comparisons lead to significant participant retention issues.

Our results also suggest new empirical and theoretical methods for studying persuasion at scale. Only recently have studies of persuasion moved beyond small scale lab studies involving simple single arguments (Habernal and Gurevych, 2016b; Habernal and Gurevych, 2016a; Tan et al., 2016). Our research also suggests new methods and tools for larger scale studies of persuasion. While care must be taken in deploying these results, studies of juries and other decision making bodies suggest that exposure to a diversity of opinions and minority views are very important to countering extremism and understanding the issues at stake (Devine et al., 2000; Ludford et al., 2004). The ability to repurpose the huge number of varied opinions available in social media sites for educational purposes could provide a novel way to expose people to a diversity of views.

References

- Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats Rule and Dogs Drool: Classifying Stance in Online Debate. In *Proc. of the ACL Workshop on Sentiment and Subjectivity*.
- N. H. Anderson. 1971. Integration theory and attitude change. *Psychological Review*, 78(3):171.
- Elisabeth André, Thomas Rist, Susanne van Mulken, Martin Klesen, and Stephan Baldes. 2000. The automated design of believable dialogues for animated presentation teams. *Embodied conversational agents*, pages 220–255.
- Emily M. Bender, Jonathan T. Morgan, Meghan Oxley, Mark Zachry, Brian Hutchinson, Alex Marin, Bin Zhang, and Mari Ostendorf. 2011. Annotating social acts: Authority claims and alignment moves in wikipedia talk pages. In *Proceedings of the Workshop on Languages in Social Media*, pages 48–57. Association for Computational Linguistics.
- Oram Biran and Owen Rambow. 2011. Identifying justifications in written dialogs. In *2011 Fifth IEEE International Conference on Semantic Computing (ICSC)*, pages 162–168.
- Filip Boltuzic and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proc. of the First Workshop on Argumentation Mining*, pages 49–58.
- Charles S. Carver, Bjorn Meyer, and Michael H. Antoni. 2000. Responsiveness to threats and incentives, expectancy of recurrence, and distress and disengagement: Moderator effects in women with early stage breast cancer. *Journal of Consulting and Clinical Psychology*, 68(6):965.
- Y. Choi and C. Cardie. 2008. Learning with compositional semantics as structural inference for sub-sentential sentiment analysis. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics.
- Alexander Conrad, and Janyce Wiebe. 2012. Recognizing arguing subjectivity and argument tags. In *Proc. of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 80–88.
- S. D. Craig, J. Sullins, A. Witherspoon, and B. Gholson. 2006. The deep-level-reasoning-question effect: The role of dialogue and deep-level-reasoning questions during vicarious learning. *Cognition and Instruction*, 24(4):565–591.
- M. F. Davies. 1998. Dogmatism and belief formation: Output interference in the processing of supporting and contradictory cognitions. *Journal of personality and social psychology*, 75(2):456.
- D. J. Devine, L. D. Clayton, B. B. Dunford, R. Seying, and J. Pryce. 2000. Jury decision making: 45 years of empirical research on deliberating groups. *Psychology, Public Policy, and Law*, 7(3):622–727.
- Barbara Di Eugenio, Nick Green, and Rajen Subba. 2013. Detecting life events in feeds from twitter. In *ICSC*, pages 274–277.
- D. M. Driscoll, S. D. Craig, B. Gholson, M. Ventura, X. Hu, and A. C. Graesser. 2003. Vicarious learning: Effects of overhearing dialog and monologue-like discourse in a virtual tutoring session. *Journal of Educational Computing Research*, 29(4):431–450.
- A. H. Eagly and S. Chaiken. 1975. An attribution analysis of the effect of communicator characteristics on opinion change: The case of communicator attractiveness. *Journal of Personality and Social Psychology*, 32(1):136.
- Hao Fang, Hao Cheng, and Mari Ostendorf. 2016. Learning latent local conversation modes for predicting community endorsement in online discussions. In *Conference on Empirical Methods in Natural Language Processing*, page 55.
- J. E. Fox Tree and S. A. Mayer. —2008—. Overhearing single and multiple perspectives. *Discourse Processes*, 45(160-179).
- J. E. Fox Tree. —1999—. Listening in on monologues and dialogues. *Discourse Processes*, 27:35–53.
- Lewis R. Goldberg. 1990. An alternative “description of personality”: The Big-Five factor structure. *Journal of Personality and Social Psychology*, 59:1216–1229.
- S. D. Gosling, P. J. Rentfrow, and W. B. Swann. 2003. A very brief measure of the big five personality domains. *Journal of Research in Personality*, 37:504–528.
- Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. page 12141223.
- Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- M. Hall, F. Eibe, G. Holms, B. Pfahringer, P. Reutemann, and I. Witten. 2005. The weka data mining software: An update. *SIGKDD Explorations*, 11(1).
- Raquel Justo, Thomas Corcoran, Stephanie M. Lukin, Marilyn Walker, and M. Inés Torres. 2014. Extracting relevant knowledge for the detection of sarcasm and nastiness in the social web. *Knowledge-Based Systems*.

- H. C. Kelman. 1961. Processes of opinion change. *Public Opinion Quarterly*, 25(1):57.
- J. Lee, F. Dineen, and J. McKendree. 1998. Supporting student discussions: it isn't just talk. *Education and Information Technologies*, 3(3):217–229.
- J. Lee. 2010. Vicarious learning from tutorial dialogue. *Sustaining TEL: From Innovation to Learning and Practice*, pages 524–529.
- Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence*, pages 185–191.
- K. Y. Liu and J. E. Fox Tree. 2011. Eavesdropping on friends and strangers: The influence of perceived familiarity on overhearer discourse comprehension. In *52nd Annual Meeting of the Psychonomics Society*.
- Zhe Liu, Yi Wang, Jalal Mahmud, Rama Akkiraju, Jerald Schoudt, Anbang Xu, and Bryan Donovan. 2016. To buy or not to buy? understanding the role of personality traits in predicting consumer behaviors. In *International Conference on Social Informatics*, pages 337–346. Springer.
- Joseph A. Luchok and James C. McCroskey. 1978. The effect of quality of evidence on attitude change and source credibility. *The Southern Speech Communication Journal*, 43:371–383.
- P. J. Ludford, D. Cosley, D. Frankowski, and L. Terveen. 2004. Think different: increasing online community participation using uniqueness and group dissimilarity. In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 631–638.
- Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. *NAACL 2013*, page 30.
- François Mairesse and Marilyn A. Walker. 2006a. Automatic recognition of personality in conversation. In *Proceedings of HLT-NAACL*.
- François Mairesse and Marilyn A. Walker. 2006b. Words mark the nerds: Computational models of personality recognition through language. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, pages 543–548.
- Traci Mann, David Sherman, and John Updegraff. 2004. Dispositional motivations and message framing: a test of the congruency hypothesis in college students. *Health Psychology*, 23(3):330.
- Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444.
- Amita Misra and Marilyn A. Walker. 2015. Topic independent identification of agreement and disagreement in social media dialogue. In *Proc. of the SIGDIAL 2013 Conference: The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.
- Amita Misra, Pranav Anand, Jean E. Fox Tree, and Marilyn Walker. 2015. Using summarization to discover argument facets in dialog. In *Proc. of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Amita Misra, Brian Ecker, Theodore Handleman, Nicolas Hahn, and Marilyn Walker. 2016a. Nlidsucsc at semeval-2016 task 6: A semi-supervised approach to detecting stance in tweets. In *Proc. of the International Workshop on Semantic Evaluation*.
- Amita Misra, Brian Ecker, and Marilyn A. Walker. 2016b. Measuring the similarity of sentential arguments in dialogue. In *Proc. of the SIGDIAL 2015 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.
- Huy V. Nguyen and Diane J. Litman. 2015. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 22–28.
- B. Nonnecke and J. Preece. 2000. Lurker demographics: Counting the silent. In *Proc. of the SIGCHI conference on Human factors in computing systems*, pages 73–80.
- W. T. Norman. 1963. Toward an adequate taxonomy of personality attributes: Replicated factor structure in peer nomination personality rating. *Journal of Abnormal and Social Psychology*, 66:574–583.
- Shereen Oraby, Lena Reed, Ryan Compton, Ellen Riloff, Marilyn Walker, and Steve Whittaker. 2015. And that's a fact: Distinguishing factual and emotional argumentation in online dialogue. In *NAACL HLT 2015 Workshop on Argument Mining*, page 116.
- Shereen Oraby, Vrindavan Harrison, Ernesto Hernandez, Lena Reed, Ellen Riloff, and Marilyn Walker. 2016. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *Proc. of the SIGDIAL 2015 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.
- Shimei Pan and Michelle Zhou. 2014. Pplum: A framework for large-scale personal persuasion. In *Proceedings of the 3rd Workshop on Data-Driven User Behavioral Modeling and Mining from Social Media*, pages 5–6. ACM.
- R. E. Petty and J. T. Cacioppo. 1986. The elaboration likelihood model of persuasion. *Advances in experimental social psychology*, 19(1):123–205.

- Richard E. Petty and John T. Cacioppo. 1988. The effects of involvement on responses to argument quantity and quality: Central and peripheral routes to persuasion. *Journal of Personality and Social Psychology*, 46(1):69–81.
- R. E. Petty, J. T. Cacioppo, and R. Goldman. 1981. Personal involvement as a determinant of argument-based persuasion. *Journal of Personality and Social Psychology*, 41(5):847.
- J. Preece, B. Nonnecke, and D. Andrews. 2004. The top five reasons for lurking: improving community experiences for everyone. *Computers in Human Behavior*, 20(2):201–223.
- Elahe Rahimtoroghi, Thomas Corcoran, Reid Swanson, Marilyn A. Walker, Kenji Sagae, and Andrew S. Gordon. 2014. Minimal narrative annotation schemes and their applications. In *7th Workshop on Intelligent Narrative Technologies*.
- Elahe Rahimtoroghi, Ernesto Hernandez, and Marilyn A. Walker. 2016. Learning fine-grained knowledge about contingent relations between everyday events. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 350.
- Chris Reed and Glenn Rowe. 2004. Araucaria: Software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools*, 13(04):961–979.
- Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *COLING*, pages 1501–1510.
- S. Stoyanchev and P. Piwek. 2010. Harvesting reusable high-level rules for expository dialogue generation. In *Proc. of the 6th International Natural Language Generation Conference*, pages 145–154.
- Rajen Subba and Barbara Di Eugenio. 2007. Automatic discourse segmentation using neural networks. In *Proc. of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, pages 189–190.
- S. V. Suzuki and S. Yamada. 2004. Persuasion through overheard communication by life-like agents. In *Intelligent Agent Technology, 2004. (IAT 2004). Proc. . IEEE/WIC/ACM International Conference on*, pages 225–231. IEEE.
- Reid Swanson, Stephanie Lukin, Luke Eisenberg, Thomas Chase Corcoran, and Marilyn A. Walker. 2014. Getting reliable annotations for sarcasm in online dialogues. In *Language Resources and Evaluation Conference, LREC 2014*.
- Reid Swanson, Brian Ecker, and Marilyn Walker. 2015. Argument mining: Extracting arguments from online dialogue. In *Proc. of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 217–226.
- Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, pages 613–624. International World Wide Web Conferences Steering Committee.
- Avril Thorne and V. Nam. 2009. The storied construction of personality. In Kitayama S. and Cohen D., editors, *The Cambridge Handbook of Personality Psychology*, pages 491–505.
- V. G. Vydiswaran, ChengXiang Zhai, Dan Roth, and Peter Pirolli. 2012. Biastrust: Teaching biased users about controversial topics. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1905–1909. ACM.
- Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. 2016. Using argument mining to assess the argumentation quality of essays. In *Proceedings of the 26th International Conference on Computational Linguistics*.
- Marilyn Walker, Pranav Anand, Rob Abbott, Jean E. Fox Tree, Craig Martell, and Joseph King. 2012a. That’s your evidence?: Classifying stance in online political debate. *Decision Support Sciences*.
- Marilyn Walker, Pranav Anand, Robert Abbott, and Richard Grant. 2012b. Stance classification using dialogic properties of persuasion. In *Meeting of the North American Association for Computational Linguistics. NAACL-HLT12*.
- Marilyn A. Walker, Jean E. Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012c. A corpus for research on deliberation and debate. In *LREC*, pages 812–817.
- S. Whittaker. 1996. Talking to strangers: an evaluation of the factors affecting electronic collaboration. In *Proc. of the 1996 ACM conference on Computer supported cooperative work*, pages 409–418.
- Nick Yee, Nicolas Ducheneaut, Les Nelson, and Peter Likarish. 2011. Introverted elves & conscientious gnomes: the expression of personality in world of warcraft. In *Proc. of the 2011 annual conference on Human factors in computing systems, CHI ’11*, pages 753–762, New York, NY, USA.

A Language-independent and Compositional Model for Personality Trait Recognition from Short Texts

Fei Liu^{♣*}, Julien Perez[♡] and Scott Nowson^{♣*}

[♣]The University of Melbourne, Victoria, Australia

[♡]Xerox Research Centre Europe, Grenoble, France

[♣]Accenture Centre for Innovation, Dublin, Ireland

fliu3@student.unimelb.edu.au

julien.perez@xrce.xerox.com

scott.nowson@accenture.com

Abstract

There have been many attempts at automatically recognising author personality traits from text, typically incorporating linguistic features with conventional machine learning models, e.g. linear regression or Support Vector Machines. In this work, we propose to use deep-learning-based models with atomic features of text – the characters – to build hierarchical, vectorial word and sentence representations for the task of trait inference. On a corpus of tweets, this method shows state-of-the-art performance across five traits and three languages (English, Spanish and Italian) compared with prior work in author profiling. The results, supported by preliminary visualisation work, are encouraging for the ability to detect complex human traits.

1 Introduction

Deep-learning methods are becoming increasingly applied to problems in the area of Natural Language Processing (NLP) (Manning, 2016). Such techniques can be applied to tasks such as part-of-speech-tagging (Ling et al., 2015; Huang et al., 2015) and sentiment analysis (Socher et al., 2013; Kalchbrenner et al., 2014; Kim, 2014). At their core, these tasks can be seen as learning representations of language at different levels. Our work reported here is no different, though we choose a less commonplace level of representation – rather than the text itself, we focus on the author behind the text. Automatically modelling individuals from their language use is a task founded on the long-standing understanding that language use is influenced by sociodemographic characteristics

such as gender and personality (Tannen, 1990; Pennebaker et al., 2003). The study of personality traits in particular is considered reliable as such traits are generally temporally stable (Matthews et al., 2003). As such, our ability to model our target – the author – is enriched by the acquisition of more data over time.

The volume of literature on computational personality recognition, and its broader applications, has grown steadily over the last decade. There have also been a number of dedicated workshops (Celli et al., 2014; Tkalčič et al., 2014) and shared tasks (Rangel et al., 2015) on the topic occurring in recent years. A significant portion of this prior literature has used some variation of enriched bag-of-words; e.g. the Open vocabulary approach (Schwartz et al., 2013). This is, theoretically speaking, entirely understandable as study of the relationship between word use and traits has delivered significant insight into human behaviour (Pennebaker et al., 2003). Language has been represented at a number of different levels in this work such as syntactic, semantic, and categorical - for example the psychologically-derived lexica of the Linguistic Inquiry and Word Count (LIWC) tool (Pennebaker et al., 2015).

These bag-of-linguistic-features approaches, however, require considerable feature engineering effort. In addition, many linguistic techniques and features are language-dependent, e.g. LIWC (Pennebaker et al., 2015), making the adaptation of models to multi-lingual scenarios more challenging. Another concern is a common assumption that these features, like the traits with which their use correlates, are similarly stable: the same language features always indicate the same traits. However, this is not the case: the relationship between language and personality is not consistent across all forms of communication, it is more complex (Nowson and Gill, 2014).

*Work carried out at Xerox Research Centre Europe

In order to address these challenges we propose a novel feature-engineering-free, deep-learning-based approach to the problem of personality trait recognition, enabling the model to work in various languages without the need to create language-specific linguistic features. We frame the problem as a supervised sequence regression task, taking only the joint atomic representation of the text: hierarchically on the character and word level. In this work, we focus on short texts. As pointed out by Han and Baldwin (2011), classification of such texts can often be challenging for even state-of-the-art BoW based approaches, which is, in part, caused by the noisy nature of such data. In this work, we address this by proposing a novel hierarchical neural network architecture, free of feature engineering and, once trained, capable of inferring personality across five traits and three languages.

The paper is structured as follows: we consider previous approaches to computational personality recognition, including those few which have a deep-learning component, and subsequently describe our model. We report two sets of experiments, the first to demonstrate the effectiveness of the model in inferring personality compared to the current state-of-the-art models, while the second reports analysis against other feature-engineering-free models. In both settings, the proposed model achieves state-of-the-art performance across five personality traits and three languages.

2 Related Work

Early work in computational personality recognition (Argamon et al., 2005; Nowson and Oberlander, 2006) were mainly SVM-based approaches, relying on syntactic and lexical features. A decade later, still “most” participants of the PAN 2015 Author Profiling task use SVM with feature engineering, according to the organisers (Rangel et al., 2015). Ensemble methods have been proposed (Verhoeven et al., 2013), but the base model was still SVM – the ensemble came from the combination of data from different sources as opposed to models. Data – not just text – labelled with personality traits is sparse (Nowson and Gill, 2014) and most work has focused on reporting novel feature sets rather than techniques. In the PAN task alone¹, there were features, in the form of surface forms of text, present on multiple levels of

¹Due to space consideration we are unable to cite the individual works.

language representation, ranging from lexical features (e.g., word, lemma and character n-grams) to syntactic ones (e.g., POS tags and dependency relations). Some, on the other hand, focused on feature curation, analysing the correlation between personality and the use of punctuation and emoticon, along with the use of the topic modelling method: latent semantic analysis. In addition, external resources, such as LIWC (Pennebaker et al., 2015), constructed over 20 years of psychology-based feature engineering, are another often-used set of features. When applied to tweets, however, LIWC requires further cleaning of the data (Kreindler, 2016).

Approaches to personality trait recognition based on deep-learning are few, which is not surprising given the relatively small scale of the data sets used. Kalghatgi et al. (2015) employed a neural network based approach. In this model, a Multilayer Perceptron (MLP) takes as input a number of carefully hand-crafted syntactic and social behavioural features from each user and attempts to predict a label for each of the 5 personality traits. However, the authors reported neither evaluation of this work, nor details of the dataset. The work of Su et al. (2016), on the other hand, employs a Recurrent Neural Network (RNN), exploiting the turn-taking nature of conversation for personality trait prediction. In their work, the RNN processes the evolution of a dialogue over time, taking as input LIWC-based and grammatical features, the output of which is then fed into the classifier for the prediction of personality trait scores of each participant of the conversations. It should be noted that both works take manually-designed features, heavily relying on domain expertise. Also, the focus is on the prediction of trait scores on the author level based on modelling all available text from a user. In contrast, not only does our approach infer the personality of a user given a collection of short texts, it is also flexible enough to predict trait scores from a single short text, arguably a more challenging task considering the limited amount of information.

In Section 3.2, we propose a model inspired by the work of Ling et al. (2015) where representations are hierarchically constructed from characters to words. This is based on the assumption that character sequences are syntactically and semantically informative of the words they compose. Their model – a widely used RNN vari-

ant Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) – learns how to construct word embeddings via its constituent characters. When applied to the tasks of language modelling and part-of-speech tagging, the model successfully improves the accuracy upon traditional baselines, performing particularly well in morphologically rich languages. Not only does the model achieve better performance on both tasks, it also has significantly fewer parameters to learn compared to a word look-up table based approach as the number of different characters is much fewer than the number of different words in a vocabulary. Moreover, the model is able to generate a sensible representation for previously unseen words. Following this, Yang et al. (2016) took it further to the document level, introducing Hierarchical Attention Networks where two bi-directional Gated Recurrent Units (GRUs) are used to process the sequence of words and then sentences respectively with the sentence-level GRU taking as input the output of the word-level GRU and returning the representation of the document. While Yang et al. (2016) describe a means to hierarchically build representations of documents from words to sentences and eventually to documents (Word to Sentence to Document, W2S2D), the work of (Ling et al., 2015) is positioned at a more fine-grained level, incorporating information from the sequence of characters (Character to Word, C2W). In this paper, the model we propose is situated between C2W and W2S2D – connecting characters, words and sentences, and ultimately personality traits (Character to Word to Sentence for Personality Trait, C2W2S4PT).

3 Model

In this section, we first identify some current issues and limitations associated with a commonly-used approach to representing text to motivate our methodology. Then, we detail the elements of the proposed language-independent, compositional model to address the problems.

3.1 Issues with the Current Approach

When applying deep learning models to NLP problems, a commonly used approach is to map words to dense real-valued vectors in a low-dimensional space with word lookup tables. Typically, for this approach to work well, one needs to train on a large corpus in an unsupervised fash-

ion, e.g. `word2vec` (Mikolov et al., 2013a; Mikolov et al., 2013b) and `GloVe` (Pennington et al., 2014), in order to obtain a sensible set of embeddings. While this approach has demonstrated its strong capabilities of capturing syntactic and semantic information and been successfully applied to a number of NLP tasks (Socher et al., 2013; Kalchbrenner et al., 2014; Kim, 2014), as identified by Ling et al. (2015), there are two practical problems with it. First, given that language is flexible, previously unseen words are bound to occur regardless of the size of the unsupervised training corpus. This problem is even more pronounced when dealing with user-generated text, such as from social media (e.g. Twitter and Facebook) due to the noisy nature of such platforms – e.g. typos, ad hoc acronyms and abbreviations, phonetic substitutions, and even meaningless strings (Han and Baldwin, 2011). One simple solution is to represent all unknown words with a special UNK vector. However, this sacrifices the meaning of the unknown word which may be critical. Moreover, it is unable to generalise to made up words, for instance, *beautification*, despite the constituent words *beautiful* and *-ification* having been observed. Second, the large number of parameters for a model to learn tends to cause overfitting. Suppose a vector of d dimensions is used to represent each word and the word lookup table is therefore of size $d \times |V|$ where $|V|$ is the vocabulary size, which normally scales to the order of hundreds and thousands. Again, this problem is particularly serious in noisier domain.

In author profiling, a large array of character-based features have been explored and shown to be effective for trait inference, such as character flooding (Nowson et al., 2015; Giménez et al., 2015), character n-grams (González-Gallardo et al., 2015; Sulea and Dichiu, 2015), and emoticons (Nowson et al., 2015; Palomino-Garibay et al., 2015). This motivates our proposed model, described in the next section, where character, word and sentence representations are hierarchically constructed, independent of a specific language and capable of harnessing personality-sensitive signals buried as deep as the character level.

3.2 Character to Word to Sentence for Personality Traits

We address the identified problems in Section 3.1 by extending the compositional character to word

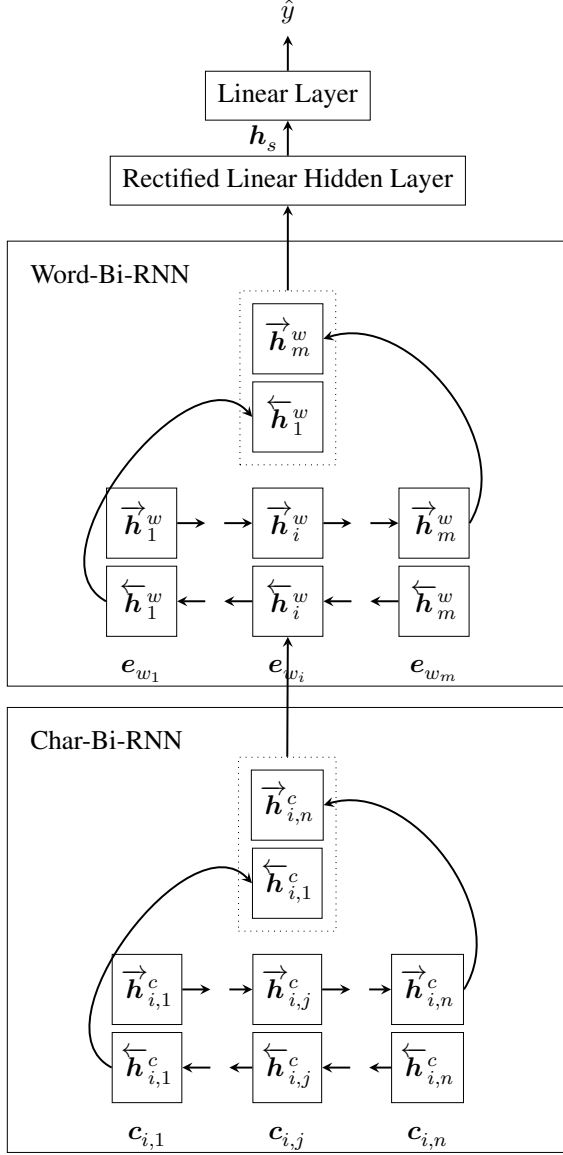


Figure 1: Illustration of the C2W2S4PT model. Dotted boxes indicate concatenation.

model (C2W) (Ling et al., 2015) wherein the constituent characters of each word is taken as input to a character-level bi-directional RNN (Char-Bi-RNN) to construct the representation of the word. A sentence is in turn represented, via another bi-directional RNN operating at the word level (Word-Bi-RNN), by the concatenation of the last and first hidden states of the forward and backward Word-RNNs respectively. Ultimately, a feedforward neural network predicts a scalar for a specific personality trait based on the input of the representation of a sentence. Given the hierarchical nature of the model, we name it C2W2S4PT (Character to Word to Sentence for Personality Traits)

depicted in Figure 1. The formal definition is provided as follows where we illustrate C2W2S4PT with an example in which a sentence s is seen as a sequence of words $\{w_1, w_2, \dots, w_i, \dots, w_m\}$ and a word w_i is in turn a sequence of characters $c_{i,j}$ whose embedding is denoted: $c_{i,j}$. Next, the Char-Bi-RNN takes as input the sequence of character embeddings $\{c_{i,1}, \dots, c_{i,n}\}$ (assuming w_i is comprised of n characters) to construct the representation of word w_i , resulting in the word embedding e_{w_i} . Here, the recurrent unit we employ in the Bi-RNNs is GRU as suggested by recent studies that GRUs achieve comparable, if not better, results to LSTM but are less demanding computationally (Chung et al., 2014; Kumar et al., 2015; Jozefowicz et al., 2015).² Concretely, the character embeddings are processed by the Char-Bi-RNN using the following:

$$\vec{z}_{i,j}^c = \sigma(\vec{W}_z^c c_{i,j} + \vec{U}_{hz}^c \vec{h}_{i,j-1}^c + \vec{b}_z^c) \quad (1)$$

$$\vec{r}_{i,j}^c = \sigma(\vec{W}_r^c c_{i,j} + \vec{U}_{hr}^c \vec{h}_{i,j-1}^c + \vec{b}_r^c) \quad (2)$$

$$\vec{h}_{i,j}^c = f(\vec{W}_h^c c_{i,j} + \vec{r}_{i,j}^c \odot \vec{U}_{hh}^c \vec{h}_{i,j-1}^c + \vec{b}_h^c) \quad (3)$$

$$\vec{h}_{i,j}^c = \vec{z}_{i,j}^c \odot \vec{h}_{i,j-1}^c + (1 - \vec{z}_{i,j}^c) \odot \vec{h}_{i,j}^c \quad (4)$$

where \odot is the element-wise product, σ the sigmoid function, f the hyperbolic tangent function \tanh , $\vec{W}_z^c, \vec{W}_r^c, \vec{W}_h^c, \vec{U}_{hz}^c, \vec{U}_{hr}^c, \vec{U}_{hh}^c$ are the parameter matrices to learn, and $\vec{b}_z^c, \vec{b}_r^c, \vec{b}_h^c$ the bias terms. In addition to the forward pass, the Char-Bi-RNN also processes the character sequence backwards (symbolised by $\overleftarrow{h}_{i,j}^c$) with another set of GRU weight matrices and bias terms. Note that the same character embeddings are shared across the forward and backward pass. Eventually, we represent w_i as the concatenation of the last and first hidden states of the forward and backward Char-RNNs:

$$e_{w_i} = \begin{bmatrix} \vec{h}_{i,n}^c \\ \overleftarrow{h}_{i,1}^c \end{bmatrix} \quad (5)$$

Sentence representations are built in a similar fashion to word representations with another Bi-RNN operating at the word level (Word-Bi-RNN) where e_{w_i} (for $i \in [1, n]$) once all the word repre-

²We performed additional experiments which confirmed this finding. Therefore due to space considerations, we do not report results using LSTMs here.

sentations have been constructed from their constituent characters) are processed:

$$\vec{z}_i^w = \sigma(\vec{W}_z^w e_{w_i} + \vec{U}_{hz}^w \vec{h}_{i-1}^w + \vec{b}_z^w) \quad (6)$$

$$\vec{r}_i^w = \sigma(\vec{W}_r^w e_{w_i} + \vec{U}_{hr}^w \vec{h}_{i-1}^w + \vec{b}_r^w) \quad (7)$$

$$\vec{h}_i^w = f(\vec{W}_h^w e_{w_i} + \vec{r}_i^w \odot \vec{U}_{hh}^w \vec{h}_{i-1}^w + \vec{b}_h^w) \quad (8)$$

$$\vec{h}_i^w = \vec{z}_i^w \odot \vec{h}_{i-1}^w + (1 - \vec{z}_i^w) \odot \vec{h}_i^w \quad (9)$$

where $\vec{W}_z^w, \vec{W}_r^w, \vec{W}_h^w, \vec{U}_{hz}^w, \vec{U}_{hr}^w, \vec{U}_{hh}^w$ are the parameter matrices to learn, and $\vec{b}_z^w, \vec{b}_r^w, \vec{b}_h^w$ the bias terms. The representation of the sentence is constructed, in a similar manner to how words are represented, by taking the concatenation of the last and first hidden states of the forward and backward Word-RNN:

$$e_s = \begin{bmatrix} \vec{h}_m^w \\ \vec{h}_1^w \end{bmatrix} \quad (10)$$

Lastly, the score for a particular personality trait is estimated with an MLP, taking as input the sentence embedding e_s and returning the estimated score \hat{y}_s :

$$\mathbf{h}_s = \text{ReLU}(\mathbf{W}_{eh} e_s + \mathbf{b}_h) \quad (11)$$

$$\hat{y}_s = \mathbf{W}_{hy} \mathbf{h}_s + b_y \quad (12)$$

where ReLU (REctified Linear Unit) is defined as $\text{ReLU}(x) = \max(0, x)$, $\mathbf{W}_{eh}, \mathbf{W}_{hy}$ the parameter matrices to learn, \mathbf{b}_h, b_y the bias terms, and \mathbf{h}_s the hidden representation of the MLP. All the trainable parameter/embedding matrices and bias terms are jointly optimised using *mean square error* as the objective function:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n (y_{s_i} - \hat{y}_{s_i})^2 \quad (13)$$

where y_{s_i} is the gold standard personality score of sentence s_i and θ the collection of all parameter/embedding matrices and bias terms for the model to learn. Note that no language-dependent component is present in the proposed model.

4 Experiments and Results

We evaluate our model in two settings, against models with or without feature engineering, to fully study the effectiveness of the proposed method. In the former, we compare – at the user

level – our feature-engineering-free and language-independent model with current state-of-the-art models which make much use of linguistic features. In the latter, on the other hand, we investigate the performance against other feature-engineering-free models on individual short texts. In both settings, we show that our model achieves better results across two language (English and Spanish) and is equally competitive in Italian.

4.1 Dataset and Preprocessing

The dataset we adopt in this paper is the English, Spanish and Italian data from the PAN 2015 Author Profiling task dataset (Rangel et al., 2015), collected from Twitter and consisting of 14,166 English (EN), 9,879 Spanish (ES) and 3,687 Italian (IT) tweets (from 152, 110 and 38 users respectively). Due to space constraints and the limited size of the data, the Dutch dataset is not included. Each user encompasses a set of tweets (average $n = 100$) with gold standard personality labels, the five trait labels (essentially scores between -0.5 and 0.5), calculated following the author’s self-assessment responses to the short Big 5 test, BFI-10 (Rammstedt and John, 2007) which has the most solid grounding in language and is considered to be the most widely accepted and exploited scheme for personality recognition (Poria et al., 2013).

In our experiments, we tokenise each tweet with Twokenizer (Owoputi et al., 2013) to preserve user mentions and hashtag-preceded topics. User mentions and URLs, unlike the majority of the language used in tweets, are intended for their targets, whose surface forms are deemed hardly informative. We therefore further normalise these features to single characters (e.g., `@username` \rightarrow `@`, `http://t.co/` \rightarrow `^`), limiting the risk of modelling unnecessary character usage not directly influenced by nor reflecting the personality of the author.

4.2 Evaluation Metric

As the test corpus is unavailable, withheld by the PAN 2015 organisers, k -fold cross-validation is used instead to compare the performance ($k = 5$ or 10) on the available dataset. To evaluate the performance, we measure the Root Mean Square Error (RMSE) at either the tweet or user level depending on the granularity of the task: $RMSE_{tweet} = \sqrt{\frac{\sum_{i=1}^T (y_{s_i} - \hat{y}_{s_i})^2}{T}}$ and $RMSE_{user} = \sqrt{\frac{\sum_{i=1}^U (y_{user_i} - \hat{y}_{user_i})^2}{U}}$ where y_{s_i}

and \hat{y}_{s_i} are the gold standard and predicted personality trait score of the i^{th} tweet whereas y_{user_i} and \hat{y}_{user_i} are their user-level counterparts, T and U the total numbers of tweets and users in the corpus. Note that in the dataset utilised in this work, each user is assigned a single score for a particular personality trait and every tweet collected from the same account inherits the same five personality trait assignments as its author. The predicted user level trait score is calculated: $\hat{y}_{user_i} = \frac{1}{T_i} \sum_{j=1}^{T_i} \hat{y}_{s_j}$ where T_i is the total number of tweets of $user_i$. In Section 4.3 and 4.4, the results, measured with $RMSE_{user}$ and $RMSE_{tweet}$, in the two settings, i.e. against models with and without feature-engineering, are presented respectively. Consistent with prior work in author profiling (Sulea and Dichiu, 2015; Mirkin et al., 2015; Nowson et al., 2015), we employ exactly the same evaluation metric on the same dataset to enable direct comparison.

4.3 Evaluation against State-of-the-art Models

We present the results obtained by the proposed model tested on the dataset described in Section 4.1. Note that our model is trained to predict personality trait scores, relying only on the text without any additional features. To enable direct comparison, we evaluate C2W2S4PT on the user level against current state-of-the-art models which incorporate linguistic features based on psychological studies.

For 5-fold cross-validation, we select the tied-highest ranked (in EN under evaluation conditions) amongst the PAN 2015 participants (Sulea and Dichiu, 2015) (also ranked 7th and 4th in ES and IT).³ Similarly, we choose baselines by ranking and metric reporting for 10-fold cross validation (Nowson et al., 2015) (ranked 9th, 6th and 8th in EN, ES and IT). In addition to the above works which predicted scores on text level and then averaged for each user, we also include subsequent work by (Mirkin et al., 2015) who reported results on concatenated tweets (a single document per author). Also, there is the most straightforward baseline Average Baseline assigning the average of all the scores to each user. We train C2W2S4PT with Adam (Kingma and Ba, 2014) over 100 epochs with a batch size of 32 and the fol-

³Cross-validation $RMSE_{user}$ performance is not reported for the other top system (Álvarez-Carmona et al., 2015).

lowing hyper-parameters: $\vec{h}_{i,j}^c$ and $\overleftarrow{h}_{i,j}^c \in \mathbb{R}^{256}$, $E_c \in \mathbb{R}^{50 \times |C|}$, dropout rate to the embedding output: 0.5, \vec{h}_i^w and $\overleftarrow{h}_i^w \in \mathbb{R}^{256}$, $W_{hy} \in \mathbb{R}^{256 \times 1}$, $b_y \in \mathbb{R}$, $W_{eh} \in \mathbb{R}^{512 \times 256}$, $b_h \in \mathbb{R}^{256}$. The $RMSE_{user}$ results are presented in Table 1 where EXT, STA, AGR, CON and OPN are abbreviations for Extroversion, Emotional Stability (the inverse of Neuroticism), Agreeableness, Conscientiousness and Openness respectively.

C2W2S4PT outperforms the current state of the art in EN and ES. In the 5-fold cross-validation group, C2W2S4PT demonstrates its advantages, attaining superior performance to the baselines except for CON in ES. In terms of 10-fold cross validation, the superiority of our model is even more evident, supported by the dominating performance over the two selected baselines across all personality traits and two languages. In both groups, 5 or 10-fold cross validation, not only does C2W2S4PT outperform the baseline systems, particularly significantly in the 10-fold group, it also does so without the aid of any hand-crafted features, stressing the technical soundness of C2W2S4PT.

On CON in ES, 5-fold cross-validation. We suspect that the surprisingly good performance of Sulea and Dichiu (2015) may likely be attributed to overfitting. Indeed, the performance on the test set on CON in ES is even inferior to Nowson et al. (2015), further confirming our speculation.

The superiority of C2W2S4PT is less clear in IT. This can possibly be caused by the inadequate amount of Italian data, less than 4k tweets as compared to 14k and 10k in the English and Spanish datasets, limiting the capability of C2W2S4PT to learn a reasonable model.

4.4 Evaluation against Other Feature-engineering-free Methods

While it is common practice in personality trait inference to evaluate at the user level, we also look into tweet-level performance to further study the models' capabilities at a more fine-grained level. A number of baselines, incorporating only the surface form of the text for the purpose of fair comparison, have been created to support our evaluation. First, we inherit the same Average Baseline as in Section 4.3. Next, we select two BoW-based system, Random Forest and SVM Regression, and

Lang.	k	Model	EXT	STA	AGR	CON	OPN
EN	—	Average Baseline	0.166	0.223	0.158	0.151	0.146
	5	Sulea and Dichiu (2015)	0.136	0.183	0.141	0.131	0.119
		C2W2S4PT	0.131	0.171	0.140	0.124	0.109
	10	Mirkin et al. (2015)	0.171	0.223	0.173	0.144	0.146
		Nowson et al. (2015)	0.153	0.197	0.154	0.144	0.132
	C2W2S4PT	0.130	0.167	0.137	0.122	0.109	
ES	—	Average Baseline	0.171	0.203	0.163	0.187	0.166
	5	Sulea and Dichiu (2015)	0.152	0.181	0.148	0.114	0.142
		C2W2S4PT	0.148	0.177	0.143	0.157	0.136
	10	Mirkin et al. (2015)	0.153	0.188	0.155	0.156	0.160
		Nowson et al. (2015)	0.154	0.188	0.155	0.168	0.160
	C2W2S4PT	0.145	0.177	0.142	0.153	0.137	
IT	—	Average Baseline	0.162	0.172	0.162	0.123	0.151
	5	Sulea and Dichiu (2015)	0.119	0.150	0.122	0.101	0.130
		C2W2S4PT	0.124	0.144	0.130	0.095	0.131
	10	Mirkin et al. (2015)	0.095	0.168	0.142	0.098	0.137
		Nowson et al. (2015)	0.137	0.168	0.142	0.098	0.141
	C2W2S4PT	0.118	0.147	0.128	0.095	0.127	

Table 1: $RMSE_{user}$ across five traits. **Bold** highlights best performance.

perform grid search for the best hyper-parameter setup ranging: kernel \in {linear, rbf} and $C \in$ {0.01, 0.1, 1.0, 10.0} whereas for Random Forest, the number of trees is chosen from the set {10, 50, 100, 500, 1000}.

In addition to the above conventional machine-learning-based models, we further implement two simpler RNN-based models, Bi-GRU-Char and Bi-GRU-Word, which work only on the character and word level respectively. On top of the GRUs, both Bi-GRU-Char and Bi-GRU-Word share the same MLP classifier, h_s and \hat{y}_s , as in C2W2S4PT. For training, we use the same set of hyper-parameters as described in Section 4.3 for C2W2S4PT. Similarly, we set the character and word embedding size to 50 and 256 for Bi-GRU-Char and Bi-GRU-Word respectively. Hyper-parameter fine-tuning was not performed mainly due to time constraints. We present the $RMSE_{tweet}$ of each effort, measured by 10-fold stratified cross-validation, in Table 2.

C2W2S4PT is comparable with, if not superior to, the strong baselines SVM Regression and Random Forest in EN and ES. C2W2S4PT achieves state-of-the-art results in almost every trait except for two, AGR in EN and STA in ES. It is worth noting that C2W2S4PT

generates this competitive performance, in the feature-engineering-free setting, against SVM Regression and Random Forest without exhaustive hyper-parameter fine-tuning.

C2W2S4PT achieves better performance than the RNN-based baselines in EN and ES. Compared with Bi-GRU-Word, C2W2S4PT is less prone to overfitting because of the relatively fewer parameters for the model to learn whereas Bi-GRU-Word needs to maintain a large vocabulary embedding matrix (Ling et al., 2015). In regards to Bi-GRU-Char, the success can be attributed to C2W2S4PT’s capability of coping with arbitrary words while not forgetting information due to excessive lengths as can arise from representing a text as a sequence of characters.

The performance of C2W2S4PT is inferior to Bi-GRU-Word in IT. Bi-GRU-Word achieves the best performance across all personality traits with C2W2S4PT coming in as a close second and tying in 3 traits. Apart from the inadequate amount of Italian data causing the fluctuation in performance as explained in Section 4.3, further investigation is needed to analyse the strong performance of Bi-GRU-Word.

Lang.	Model	EXT	STA	AGR	CON	OPN
EN	Average Baseline	0.163	0.222	0.157	0.150	0.147
	SVM Regression	0.148	0.196	0.148	0.140	0.131
	Random Forest	0.144	0.192	0.146	0.138	0.132
	Bi-GRU-Char	0.150	0.202	0.152	0.143	0.137
	Bi-GRU-Word	0.147	0.200	0.146	0.138	0.130
	C2W2S4PT	0.142	0.188	0.147	0.136	0.127
ES	Average Baseline	0.171	0.204	0.163	0.187	0.165
	SVM Regression	0.158	0.190	0.157	0.171	0.152
	Random Forest	0.159	0.195	0.157	0.177	0.158
	Bi-GRU-Char	0.163	0.195	0.158	0.178	0.155
	Bi-GRU-Word	0.159	0.192	0.154	0.173	0.154
	C2W2S4PT	0.158	0.191	0.153	0.168	0.150
IT	Average Baseline	0.164	0.171	0.164	0.125	0.153
	SVM Regression	0.141	0.159	0.145	0.113	0.141
	Random Forest	0.140	0.161	0.140	0.111	0.147
	Bi-GRU-Char	0.149	0.163	0.153	0.117	0.146
	Bi-GRU-Word	0.135	0.156	0.140	0.109	0.141
	C2W2S4PT	0.139	0.156	0.143	0.109	0.141

Table 2: $RMSE_{tweet}$ across five traits level. **Bold** highlights best performance.

4.5 Visualisation

In order to investigate the features automatically learned by the models, we select C2W2S4PT trained on a single personality trait (EXT) and visualise the 2D PCA (Tipping and Bishop, 1999) scatter plot of the representations of the sentences.⁴ As examples, we randomly select 100 tweets, 50 each from either extreme of the EXT spectrum - Extraversion being selected for this exercise as it is the most commonly studied and well understood trait. The text representations are automatically constructed by C2W2S4PT, with the resultant plot presented in Figure 2. Here, two clusters are easily identifiable, representing positive and negative Extraversion, with the former intersecting the latter. We consider three examples, highlighted in Figure 2, for discussion.

- POS7: “@username: Feeling like you’re not good enough is probably the worst thing to feel.”
- NEG3: “Being good ain’t enough lately.”
- POS20: “o.O Lovely.”

The first two examples, POS7 and NEG3, although essentially similar in terms of semantics,

⁴We also experimented with t-SNE (Van der Maaten and Hinton, 2008) but it did not produce an interpretable plot.

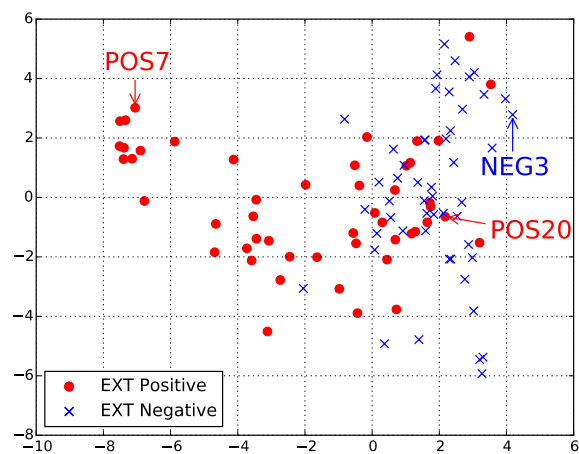


Figure 2: Scatter plot of sentence representations processed by PCA.

are placed distantly from each other at the far ends of the distribution. Despite the semantic similarities, the linguistic attributes they possess are commonly understood to be associated with Extraversion differently (Gill and Oberlander, 2002): the longer tweet, POS7, together with its use of the second person pronoun, suggests that the author is more inclusive of others while NEG3, on the other hand, is self-focused and shorter, ele-

ments signifying Introversion. The third example, POS20, while appearing to be mapped to an Introvert space, is a tweet from an Extravert. Apart from being short, POS20 incorporates the use of non-rotated, “Eastern” style emoticons (*o.O*), aspects shown to be linked to Introversion on social media (Schwartz et al., 2013). This is perhaps not the venue to consider the implications of this further, although one explanation might be that the model has uncovered a flexibility often associated with Ambiverts (Grant, 2013). However, it is worth noting that the model is capable of capturing, without feature engineering, well-understood dimensions of language.

5 Conclusion and Future Work

Overall, the results in this paper demonstrate the validity of our methodology: not only does C2W2S4PT provide state-of-the-art results compared to previous feature-engineering-heavy works, but it also performs well when compared with other widely used strong baselines in the feature-engineering-free setting. More importantly, the lack of feature engineering enables us to adapt the same model, with zero alteration to the model itself, to other languages. To further examine this property of the proposed model, we plan to explore the TwiSty dataset (Verhoeven et al., 2016), a recently introduced corpus consisting of 6 languages and labelled with MBTI type indicators (Myers and Myers, 2010).

References

- Miguel A. Álvarez-Carmona, A. Pastor López-Monroy, Manuel Montes y Gómez, Luis Villaseñor-Pineda, and Hugo Jair Escalante. 2015. INAOE’s participation at PAN’15: Author Profiling task—Notebook for PAN at CLEF 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*.
- Shlomo Argamon, Sushant Dhawle, Moshe Koppel, and James W. Pennebaker. 2005. Lexical predictors of personality type. In *Proceedings of the 2005 Joint Annual Meeting of the Interface and the Classification Society of North America*.
- Fabio Celli, Bruno Lepri, Joan-Isaac Biel, Daniel Gatica-Perez, Giuseppe Riccardi, and Fabio Pianesi. 2014. The workshop on computational personality recognition 2014. In *Proc. ACMMM*, pages 1245–1246, Orlando, USA.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Alastair J. Gill and Jon Oberlander. 2002. Taking Care of the Linguistic Features of Extraversion. In *Proc. CogSci*, pages 363–368, Fairfax, USA.
- Maite Giménez, Delia Irazú Hernández, and Ferran Pla. 2015. Segmenting Target Audiences: Automatic Author Profiling Using Tweets—Notebook for PAN at CLEF 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*.
- Carlos E. González-Gallardo, Azucena Montes, Gerardo Sierra, J. Antonio Núñez-Juárez, Adolfo Jonathan Salinas-López, and Juan Ek. 2015. Tweets Classification Using Corpus Dependent Tags, Character and POS N-grams—Notebook for PAN at CLEF 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*.
- Adam M. Grant. 2013. Rethinking the extraverted sales ideal: The ambivert advantage. *Psychological Science* 24(6), 24(6):1024–1030.
- Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proc. ACL*, pages 368–378, Portland, Oregon, USA.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. ICML*, pages 2342–2350. JMLR Workshop and Conference Proceedings.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. ACL*, Baltimore, USA.
- Mayuri Pundlik Kalghatgi, Manjula Ramannavar, and Nandini S. Sidnal. 2015. A neural network approach to personality prediction based on the big-five model. *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, 2(8):56–63.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. EMNLP*, Doha, Qatar.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

- Jon Kreindler. 2016. Twitter psychology analyzer api and sample code. <http://www.receptiviti.ai/blog/twitter-psychology-analyzer-api-and-sample-code/>. Accessed: 2016-09-30.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.
- Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fernandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. EMNLP*, pages 1520–1530, Lisbon, Portugal.
- Christopher D Manning. 2016. Computational linguistics and deep learning. *Computational Linguistics*.
- Gerald Matthews, Ian J. Deary, and Martha C. White-man. 2003. *Personality Traits*. Cambridge University Press, second edition. Cambridge Books Online.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. ICLR*, Scottsdale, USA.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*, pages 3111–3119, Stateline, USA.
- Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proc. EMNLP*, pages 1102–1108, Lisbon, Portugal.
- Isabel Myers and Peter Myers. 2010. *Gifts differing: Understanding personality type*. Nicholas Brealey Publishing.
- Scott Nowson and Alastair J. Gill. 2014. Look! Who’s Talking? Projection of Extraversion Across Different Social Contexts. In *Proceedings of WCPRI4, Workshop on Computational Personality Recognition at ACMM (22nd ACM International Conference on Multimedia)*.
- Scott Nowson and Jon Oberlander. 2006. The Identity of Bloggers: Openness and gender in personal weblogs. In *AAAI Spring Symposium, Computational Approaches to Analysing Weblogs*.
- Scott Nowson, Julien Perez, Caroline Brun, Shachar Mirkin, and Claude Roux. 2015. XRCE Personal Language Analytics Engine for Multilingual Author Profiling. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*.
- Olutobi Owoputi, Brendan O’Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. NAACL*, pages 380–390, Atlanta, USA.
- Alonso Palomino-Garibay, Adolfo T. Camacho-González, Ricardo A. Fierro-Villaneda, Irazú Hernández-Farias, Davide Buscaldi, and Ivan V. Meza-Ruiz. 2015. A Random Forest Approach for Authorship Profiling—Notebook for PAN at CLEF 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*.
- James W Pennebaker, Kate G Niederhoffer, and Matthias R Mehl. 2003. Psychological aspects of natural language use: Our words, our selves. *Annual Review of Psychology*, 54:547–577.
- J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn. 2015. The development and psychometric properties of LIWC2015. This article is published by LIWC Inc, Austin, Texas 78703 USA in conjunction with the LIWC2015 software program.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543, Doha, Qatar.
- Soujanya Poria, Alexandar Gelbukh, Basant Agarwal, Erik Cambria, and Newton Howard, 2013. *Common Sense Knowledge Based Personality Recognition from Text*, pages 484–496.
- Beatrice Rammstedt and Oliver P. John. 2007. Measuring personality in one minute or less: A 10-item short version of the big five inventory in english and german. *Journal of Research in Personality*, 41(1):203–212.
- Francisco Rangel, Fabio Celli, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd Author Profiling Task at PAN 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*, CEUR Workshop Proceedings.
- H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E P Seligman, and Lyle H Ungar. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-Vocabulary Approach. *PLOS ONE*, 8(9).
- Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*, Seattle, USA.
- Ming-Hsiang Su, Chung-Hsien Wu, and Yu-Ting Zheng. 2016. Exploiting turn-taking temporal evolution for personality trait perception in dyadic conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):733–744.

- Octavia-Maria Sulea and Daniel Dichiu. 2015. Automatic profiling of twitter users based on their tweets. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*.
- Deborah Tannen. 1990. *You Just Dont Understand: Women and Men in Conversation*. Harper Collins, New York.
- Michael E Tipping and Christopher M Bishop. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.
- Marko Tkalčič, Berardina De Carolis, Marco de Gemmis, Ante Odić, and Andrej Košir. 2014. Preface: Empire 2014. In *Proceedings of the 2nd Workshop Emotions and Personality in Personalized Services (EMPIRE 2014)*. CEUR-WS.org, July.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.
- Ben Verhoeven, Walter Daelemans, and Tom De Smedt. 2013. Ensemble Methods for Personality Recognition. In *Proceedings of WCPRI3, Workshop on Computational Personality Recognition at ICWSM13 (7th International Conference on Weblogs and Social Media)*.
- Ben Verhoeven, Walter Daelemans, and Barbara Plank. 2016. TwiSty: a multilingual twitter stylometry corpus for gender and personality profiling. In *Proc. LREC*, pages 1632–1637, Portorož, Slovenia.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proc. NAACL*, pages 1480–1489, San Diego, USA.

A Strong Baseline for Learning Cross-Lingual Word Embeddings from Sentence Alignments

Omer Levy*

University of Washington
Seattle, WA
omerlevy@gmail.com

Anders Søgaard*

University of Copenhagen
Copenhagen, Denmark
soegaard@di.ku.dk

Yoav Goldberg

Bar-Ilan University
Ramat-Gan, Israel
yoav.goldberg@gmail.com

Abstract

While cross-lingual word embeddings have been studied extensively in recent years, the qualitative differences between the different algorithms remain vague. We observe that whether or not an algorithm uses a particular feature set (sentence IDs) accounts for a significant performance gap among these algorithms. This feature set is also used by traditional alignment algorithms, such as IBM Model-1, which demonstrate similar performance to state-of-the-art embedding algorithms on a variety of benchmarks. Overall, we observe that different algorithmic approaches for utilizing the sentence ID feature space result in similar performance. This paper draws both empirical and theoretical parallels between the embedding and alignment literature, and suggests that adding additional sources of information, which go beyond the traditional signal of bilingual sentence-aligned corpora, may substantially improve cross-lingual word embeddings, and that future baselines should at least take such features into account.

1 Introduction

Cross-lingual word embedding algorithms try to represent the vocabularies of two or more languages in one common continuous vector space. These vectors can be used to improve monolingual word similarity (Faruqui and Dyer, 2014) or support cross-lingual transfer (Gouws and Søgaard, 2015). In this work, we focus on the second (cross-lingual) aspect of these embeddings, and try to determine what makes some embedding approaches better than others *on a set of*

translation-oriented benchmarks. While cross-lingual word embeddings have been used for a variety of cross-lingual transfer tasks, we prefer evaluating on translation-oriented benchmarks, rather than across specific NLP tasks, since the translation setting allows for a cleaner examination of cross-lingual similarity. Another important delineation of this work is that we focus on algorithms that rely on *sentence-aligned data*; in part, because these algorithms are particularly interesting for low-resource languages, but also to make our analysis and comparison with alignment algorithms more focused.

We observe that the top performing embedding algorithms share the same underlying feature space – sentence IDs – while their different algorithmic approaches seem to have a negligible impact on performance. We also notice that several statistical alignment algorithms, such as IBM Model-1 (Brown et al., 1993), operate under the same data assumptions. Specifically, we find that using the translation probabilities learnt by Model-1 as the cross-lingual similarity function (in place of the commonly-used cosine similarity between word embeddings) performs on-par with state-of-the-art cross-lingual embeddings on word alignment and bilingual dictionary induction tasks. In other words, as long as the similarity function is based on the sentence ID feature space and the embedding/alignment algorithm itself is not too naïve, the actual difference in performance between different approaches is marginal.

This leads us to revisit another statistical alignment algorithm from the literature that uses the same sentence-based signal – the Dice aligner (Och and Ney, 2003). We first observe that the vanilla Dice aligner is significantly outperformed by the Model-1 aligner. We then recast Dice as the dot-product between two word vectors (based on the sentence ID feature space), which allows us to

*These authors contributed equally to this work.

generalize it, resulting in an embedding model that is as effective as Model-1 and other sophisticated state-of-the-art embedding methods, but takes a fraction of the time to train.

Existing approaches for creating cross-lingual word embeddings are typically restricted to training bilingual embeddings, mapping exactly two languages onto a common space. We show that our generalization of the Dice coefficient can be augmented to jointly train *multi*-lingual embeddings for any number of languages. We do this by leveraging the fact that the space of sentence IDs is shared among all languages in the parallel corpus; the verses of the Bible, for example, are identical across all translations. Introducing this multi-lingual signal shows a significant performance boost, which eclipses the variance in performance among pre-existing embedding algorithms.

Contributions We first establish the importance of the sentence ID feature space for cross-lingual word embedding algorithms through experiments across several translation-oriented benchmarks. We then compare cross-lingual word embedding algorithms to traditional word alignment algorithms that also rely on sentence ID signals. We show that a generalization of one of these, the Dice aligner, is a very strong baseline for cross-lingual word embedding algorithms, performing better than several state-of-the-art algorithms, especially when exploiting a multi-lingual signal. Our code and data are publicly available.¹

2 Background: Cross-lingual Embeddings

Previous approaches to cross-lingual word embeddings can be divided into three categories, according to assumptions on the training data. The first category assumes *word-level alignments*, in the form of bilingual dictionaries (Mikolov et al., 2013a; Xiao and Guo, 2014) or automatically produced word alignments (Klementiev et al., 2012; Zou et al., 2013; Faruqui and Dyer, 2014). Sizable bilingual dictionaries are not available for many language pairs, and the quality of automatic word alignment greatly affects the quality of the embeddings. It is also unclear whether the embedding process provides significant added value beyond the initial word alignments (Zou et al., 2013). We

¹bitbucket.org/omerlevy/xling_embeddings

therefore exclude these algorithms for this study, also in order to focus our analysis and make the comparison with traditional word alignment algorithms more straightforward.

The second category makes a much weaker assumption, *document-level alignments*, and uses comparable texts in different languages (not necessarily translations) such as Wikipedia articles or news reports of the same event. Algorithms in this category try to leverage massive amounts of data to make up for the lack of lower-level alignments (Søgaard et al., 2015; Vulić and Moens, 2016).

Algorithms in the third category take the middle ground; they use *sentence-level alignments*, common in legal translations and religious texts. Also known as “parallel corpora”, sentence-aligned data maps each sentence (as a whole) to its translation. We focus on this third category, because it does not require the strict assumption of word-aligned data (which is difficult to obtain), while still providing a cleaner and more accurate signal than document-level alignments (which have been shown, in monolingual data, to capture mainly syntagmatic relations (Sahlgren, 2006)). In §6, we provide evidence to the hypothesis that sentence-aligned data is indeed far more informative than document-aligned data.

Algorithms that rely on sentence-aligned data typically create intermediate sentence representations from each sentence’s constituent words. Hermann and Blunsom (2014) proposed a deep neural model, BiCVM, which compared the two sentence representations at the final layer, while Chandar et al. (2014) proposed a shallower autoencoder-based model, representing both source and target language sentences as the same intermediate sentence vector. Recently, a simpler model, BilBOWA (Gouws et al., 2015), showed similar performance *without* using a hidden sentence-representation layer, giving it a dramatic speed advantage over its predecessors. BilBOWA is essentially an extension of skip-grams with negative sampling (SGNS) (Mikolov et al., 2013b), which simultaneously optimizes each word’s similarity to its inter-lingual context (words that appeared in the aligned target language sentence) and its intra-lingual context (as in the original monolingual model). Luong et al. (2015) proposed a similar SGNS-based model over the same features.

We study which factors determine the success of cross-lingual word embedding algorithms

that use sentence-aligned data, and evaluate them against baselines from the statistical machine translation literature that incorporate the same data assumptions. We go on to generalize one of these, the Dice aligner, showing that one variant is a much stronger baseline for cross-lingual word embedding algorithms than standard baselines.

Finally, we would like to point out the work of Upadhyay et al. (2016), who studied how different data assumptions affect embedding quality in both monolingual and cross-lingual tasks. Our work focuses on one specific data assumption (sentence-level alignments) and only on cross-lingual usage. This more restricted setting allows us to: (a) compare embeddings to alignment algorithms, (b) decouple the feature space from the algorithm, and make a more specific observation about the contribution of each component to the end result. In that sense, our findings complement those of Upadhyay et al. (2016).

3 Which Features Make Better Cross-lingual Embeddings?

We group state-of-the-art cross-lingual embedding algorithms according to their feature sets, and compare their performance on two cross-lingual benchmarks: word alignment and bilingual dictionary induction. In doing so, we hope to learn which features are more informative.

3.1 Features of Sentence-aligned Data

We observe that cross-lingual embeddings typically use parallel corpora in one of two ways:

Source + Target Language Words Each word w is represented using all the other words that appeared with it in the same sentence (source language words) *and* all the words that appeared in target language sentences that were aligned to sentences in which the word w appeared (target language words). This representation also stores the number of times each pair of word w and feature (context) word f co-occurred.

These features are analogous to the ones used by Vulić and Moens (2016) for document-aligned data, and can be built in a similar manner: create a pseudo-bilingual sentence from each aligned sentence, and for each word in question, consider all the other words in this sentence as its features. BilBOWA (Gouws et al., 2015) also uses a similar set of features, but restricts the source language words to those that appeared within a certain distance

from the word in question, and defines a slightly different interaction with target language words.

Sentence IDs Here, each word is represented by the set of sentences in which it appeared, indifferent to the number of times it appeared in each one. This feature set is also indifferent to the word ordering within each sentence. This approach is implicitly used by Chandar et al. (2014), who encode the bag-of-words representations of parallel sentences into the same vector. Thus, each word is not matched directly to another word, but rather used to create the sentence’s language-independent representation. Sjøgaard et al. (2015) use similar features, document IDs, for leveraging comparable Wikipedia articles in different languages. In §6 we show that when using sentence IDs, even a small amount of sentence-aligned data is more powerful than a huge amount of comparable documents.

3.2 Experiment Setup

Algorithms We use the four algorithms mentioned in §3.1: BilBOWA (Gouws et al., 2015), BWE-SkipGram (Vulić and Moens, 2016), Bilingual Autoencoders (Chandar et al., 2014), and Inverted Index (Sjøgaard et al., 2015). While both BWE-SkipGram and Inverted Index were originally trained on document-aligned data, in this work, we apply them to sentence-aligned data.

Data Christodouloupoulos and Steedman (2015) collected translations of the Bible (or parts of it) in over 100 languages, naturally aligned by book, chapter, and verse (31,102 verses in total).² This corpus allows us to evaluate methods across many different languages, while controlling for the training set’s size. The corpus was decapitalized and tokenized using white spaces after splitting at punctuation.

Benchmarks We measure the quality of each embedding using both manually annotated word alignment datasets and bilingual dictionaries. We use 16 manually annotated word alignment datasets – Hansards³ and data from four other sources (Graca et al., 2008; Lambert et al., 2005; Mihalcea and Pedersen, 2003; Holmqvist and Ahrenberg, 2011; Cakmak et al., 2012) – as well as 16 bilingual dictionaries from Wiktionary.

²homepages.inf.ed.ac.uk/s0787820/bible/

³www.isi.edu/natural-language/download/hansard/

In the word alignment benchmark, each word in a given source language sentence is aligned with the most similar target language word from the target language sentence – this is exactly the same greedy decoding algorithm that is implemented in IBM Model-1 (Brown et al., 1993). If a source language word is out of vocabulary, it is not aligned with anything, whereas target language out-of-vocabulary words are given a default minimal similarity score, and never aligned to any candidate source language word in practice. We use the inverse of alignment error rate (1-AER) as described in Koehn (2010) to measure performance, where higher scores mean better alignments.

High quality, freely available, manually annotated word alignment datasets are rare, especially for non-European languages. We therefore include experiments on bilingual dictionary induction. We obtain bilingual dictionaries from Wiktionary for five non-Indo-European languages, namely: Arabic, Finnish, Hebrew, Hungarian, and Turkish (all represented in the Edinburgh Bible Corpus). We emphasize that unlike most previous work, we experiment with finding translation equivalents of all words and do not filter the source and target language words by part of speech. We use precision-at-one (P@1), essentially selecting the closest target-language word to the given source-language word as the translation of choice. This often means that 100% precision is unattainable, since many words have multiple translations.

Hyperparameters Levy et al. (2015) exposed a collection of hyperparameters that affect the performance of monolingual embeddings. We assume that the same is true for cross-lingual embeddings, and use their recommended settings across all algorithms (where applicable). Specifically, we used 500 dimensions for every algorithm, context distribution smoothing with $\alpha = 0.75$ (applicable to BilBOWA and BWE-SkipGram), the symmetric version of SVD (applicable to Inverted Index), and run iterative algorithms for 100 epochs to ensure convergence (applicable to all algorithms except Inverted Index). For BilBOWA’s monolingual context window, we used the default of 5. Similarity is always measured by the vectors’ cosine. Most importantly, we use a shared vocabulary, consisting of every word that appeared at least twice in the corpus (tagged with language ID). While hyperparameter tuning could admittedly affect results, we rarely have data for reliably

tuning hyperparameters for truly low-resource languages.

3.3 Results

Table 1 shows that the two algorithms based on the sentence-ID feature space perform consistently better than those using source+target words. We suspect that the source+target feature set might be capturing more information than is actually needed for translation, such as syntagmatic or topical similarity between words (e.g. “dog” \sim “kennel”). This might be distracting for cross-lingual tasks such as word alignment and bilingual dictionary induction. Sentence ID features, on the other hand, are simpler, and might therefore contain a cleaner translation-oriented signal.

It is important to state that, in absolute terms, these results are quite poor. The fact that the best inverse AER is around 50% calls into question the ability to actually utilize these embeddings in a real-life scenario. While one may suggest that this is a result of the small training dataset (Edinburgh Bible Corpus), previous work (e.g. (Chandar et al., 2014)) used an even smaller dataset (the first 10K sentences in Europarl (Koehn, 2005)). To ensure that our results are not an artifact of the Edinburgh Bible Corpus, we repeated our experiments on the full Europarl corpus (180K sentences) for a subset of languages (English, French, and Spanish), and observed similar trends. As this is a comparative study focused on analyzing the qualitative differences between algorithms, we place the issue of low absolute performance aside for the moment, and reopen it in §5.4.

4 Comparing Cross-lingual Embeddings with Traditional Alignment Methods

Sentence IDs are not unique to modern embedding methods, and have been used by statistical machine translation from the very beginning. In particular, the Dice coefficient (Och and Ney, 2003), which is often used as a baseline for more sophisticated alignment methods, measures the cross-lingual similarity of words according to the number of aligned sentences in which they appeared. IBM Model-1 (Brown et al., 1993) also makes exactly the same data assumptions as other sentence-ID methods. It therefore makes sense to use Dice similarity and the translation probabilities derived from IBM Model-1 as baselines for cross-lingual embeddings that use sentence IDs.

			Source+Target Words		Sentence IDs	
			BilBOWA	BWE SkipGram	Bilingual Autoencoders	Inverted Index
Word Alignment (1-AER)	GRAÇA	en fr	.3653	.3538	.4376	.3499
		fr en	.3264	.3676	.4488	.3995
		en es	.2723	.3156	.5000	.3443
		es en	.2953	.3740	.5076	.4545
		en pt	.3716	.3983	.4449	.3263
		pt en	.3949	.4272	.4474	.3902
	HANSARDS	en fr	.3189	.3109	.4083	.3336
		fr en	.3206	.3314	.4218	.3749
	LAMBERT	en es	.1576	.1897	.2960	.2268
		es en	.1617	.2073	.2905	.2696
	MIHALCEA	en ro	.1621	.1848	.2366	.1951
		ro en	.1598	.2042	.2545	.2133
HOLMQVIST	en sv	.2092	.2373	.2746	.2357	
	sv en	.2121	.2853	.2994	.2881	
CAKMAK	en tr	.1302	.1547	.2256	.1731	
	tr en	.1479	.1571	.2661	.2665	
Dictionary Induction (P@1)	WIKTIONARY	en fr	.1096	.2176	.2475	.3125
		fr en	.1305	.2358	.2762	.3466
		en es	.0630	.1246	.2738	.3135
		es en	.0650	.1399	.3012	.3574
		en pt	.1384	.3869	.3281	.3866
		pt en	.1573	.4119	.3661	.4190
		en ar	.0385	.1364	.0995	.1364
		ar en	.0722	.2408	.1958	.2825
		en fi	.0213	.1280	.0887	.1367
		fi en	.0527	.1877	.1597	.2477
		en he	.0418	.1403	.0985	.1284
		he en	.0761	.1791	.1701	.2179
		en hu	.0533	.2299	.1679	.2182
		hu en	.0810	.2759	.2234	.3204
		en tr	.0567	.2207	.1770	.2245
		tr en	.0851	.2598	.2069	.2835
Average*			.1640	.2505	.2856	.2867
Top 1			0	3.5	15	13.5

Table 1: The performance of four state-of-the-art cross-lingual embedding methods. * Averages across two different metrics.

From Table 2 we learn that the existing embedding methods are not really better than IBM Model-1. In fact, their average performance is even slightly lower than Model-1’s. Although Bilingual Autoencoders, Inverted Index, and Model-1 reflect entirely different algorithmic approaches (respectively: neural networks, matrix factorization, and EM), the overall difference in performance seems to be rather marginal. This suggests that the main performance factor is not the algorithm, but the feature space: sentence IDs.

However, Dice also relies on sentence IDs, yet its performance is significantly worse. We suggest that Dice uses the sentence-ID feature set naïvely, resulting in degenerate performance with respect to the other methods. In the following section, we analyze this shortcoming and show that generalizations of Dice actually do yield similar perfor-

mance Model-1 and other sentence-ID methods.

5 Generalized Dice

In this section, we show that the Dice coefficient (Och and Ney, 2003) can be seen as the dot-product between two word vectors represented over the sentence-ID feature set. After providing some background, we demonstrate the mathematical connection between Dice and word-feature matrices. We then introduce a new variant of Dice, SID-SGNS, which performs on-par with Model-1 and the other embedding algorithms. This variant is able to seamlessly leverage the multi-lingual nature of sentence IDs, giving it a small but significant edge over Model-1.

			Embeddings		Alignment Algorithms	
			Bilingual Autoencoders	Inverted Index	Dice	IBM Model-1
Word Alignment (1-AER)	GRAÇA	en fr	.4376	.3499	.3355	.4263
		fr en	.4488	.3995	.3470	.4248
		en es	.5000	.3443	.3919	.4251
		es en	.5076	.4545	.3120	.4243
		en pt	.4449	.3263	.3569	.4729
		pt en	.4474	.3902	.3598	.4712
	HANSARDS	en fr	.4083	.3336	.3614	.4360
		fr en	.4218	.3749	.3663	.4499
	LAMBERT	en es	.2960	.2268	.2057	.2400
		es en	.2905	.2696	.1947	.2443
	MIHALCEA	en ro	.2366	.1951	.2030	.2335
		ro en	.2545	.2133	.1720	.2214
	HOLMQVIST	en sv	.2746	.2357	.2435	.3405
		sv en	.2994	.2881	.2541	.3559
CAKMAK	en tr	.2256	.1731	.2285	.3154	
	tr en	.2661	.2665	.2458	.3494	
Dictionary Induction (P@1)	WIKTIONARY	en fr	.2475	.3125	.1104	.1791
		fr en	.2762	.3466	.1330	.1816
		en es	.2738	.3135	.1072	.0903
		es en	.3012	.3574	.1417	.1131
		en pt	.3281	.3866	.1384	.3779
		pt en	.3661	.4190	.1719	.4358
		en ar	.0995	.1364	.0449	.1316
		ar en	.1958	.2825	.0610	.2873
		en fi	.0887	.1367	.0423	.1340
		fi en	.1597	.2477	.0463	.2394
		en he	.0985	.1284	.0358	.1224
		he en	.1701	.2179	.0328	.2000
		en hu	.1679	.2182	.0569	.2219
		hu en	.2234	.3204	.0737	.2985
		en tr	.1770	.2245	.0406	.1985
		tr en	.2069	.2835	.0820	.3073
Average			0.2856	0.2867	0.1843	0.2922
Top 1			8	12	0	12

Table 2: The performance of embedding and alignment methods based on the sentence ID feature set.

5.1 Word-Feature Matrices

In the word similarity literature, it is common to represent words as real-valued vectors and compute their “semantic” similarity with vector similarity metrics, such as the cosine of two vectors. These word vectors are traditionally derived from sparse word-feature matrices, either by using the matrix’s rows as-is (also known as “explicit” representation) or by inducing a lower-dimensional representation via matrix factorization (Turney and Pantel, 2010). Many modern methods, such as those in word2vec (Mikolov et al., 2013b), also create vectors by factorizing word-feature matrices, only without representing these matrices explicitly.

Formally, we are given a vocabulary of words V_W and a feature space (“vocabulary of features”)

V_F . These features can be, for instance, the set of sentences comprising the corpus. We then define a matrix M of $|V_W|$ rows and $|V_F|$ columns. Each entry in M represents some statistic pertaining to that combination of word and feature. For example, $M_{w,f}$ could be the number of times the word w appeared in the document f .

The matrix M is typically processed into a “smarter” matrix that reflects the strength of association between each given word w and feature f . We present three common association metrics: L_1 row normalization (Equation (1)), Inverse Document Frequency (IDF, Equation (2)), and Pointwise Mutual Information (PMI, Equation (3)). The following equations show how to compute their respective matrices:

$$M_{w,f}^{L_1} = \frac{I(w,f)}{I(w,*)} \quad (1)$$

$$M_{w,f}^{IDF} = \log \frac{|V_F|}{I(w,*)} \quad (2)$$

$$M_{w,f}^{PMI} = \log \frac{\#(w,f) \cdot \#(*,*)}{\#(w,*) \cdot \#(*,f)} \quad (3)$$

where $\#(\cdot, \cdot)$ is the co-occurrence count function, $I(\cdot, \cdot)$ is the co-occurrence indicator function, and $*$ is a wildcard.⁴

To obtain word vectors of lower dimensionality (V_F may be huge), the processed matrix is then decomposed, typically with SVD. An alternative way to create low-dimensional word vectors without explicitly constructing M is to use the negative sampling algorithm (SGNS) (Mikolov et al., 2013b).⁵ This algorithm factorizes M^{PMI} using a weighted non-linear objective (Levy and Goldberg, 2014).

5.2 Reinterpreting the Dice Coefficient

In statistical machine translation, the Dice coefficient is commonly used as a baseline for word alignment (Och and Ney, 2003). Given sentence-aligned data, it provides a numerical measure of how likely two words – a source-language word w_s and a target-language word w_t – are each other’s translation:

$$Dice(w_s, w_t) = \frac{2 \cdot S(w_s, w_t)}{S(w_s, *) \cdot S(*, w_t)} \quad (4)$$

where $S(\cdot, \cdot)$ is the number of aligned sentences in the data where both arguments occurred.

We claim that this metric is mathematically equivalent to the dot-product of two L_1 -normalized sentence-ID word-vectors, multiplied by 2. In other words, if we use the combination of sentence-ID features and L_1 -normalization to create our word vectors, then for any w_s and w_t :

$$w_s \cdot w_t = \frac{Dice(w_s, w_t)}{2} \quad (5)$$

To demonstrate this claim, let us look at the dot-product of w_s and w_t :

$$w_s \cdot w_t = \sum_i \left(\frac{I(w_s, i)}{I(w_s, *)} \cdot \frac{I(w_t, i)}{I(w_t, *)} \right) \quad (6)$$

where i is the index of an aligned sentence. Since $I(w_s, *) = S(w_s, *)$ and $I(w_t, *) = S(*, w_t)$, and both are independent of i , we can rewrite the equation as follows:

$$w_s \cdot w_t = \frac{\sum_i I(w_s, i) \cdot I(w_t, i)}{S(w_s, *) \cdot S(*, w_t)} \quad (7)$$

⁴A function with a wildcard should be interpreted as the sum of all possible instantiations, e.g. $I(w, *) = \sum_x I(w, x)$.

⁵For consistency with prior art, we refer to this algorithm as SGNS (skip-grams with negative sampling), even when it is applied without the skip-gram feature model.

Since $I(w, i)$ is an indicator function of whether the word w appeared in sentence i , it stands to reason that the product $I(w_s, i) \cdot I(w_t, i)$ is an indicator of whether both w_s and w_t appeared in i . Ergo, the numerator of Equation (7) is exactly the number of aligned sentences in which both w_s and w_t occurred: $S(w_s, w_t)$. Therefore:

$$w_s \cdot w_t = \frac{S(w_s, w_t)}{S(w_s, *) \cdot S(*, w_t)} = \frac{Dice(w_s, w_t)}{2} \quad (8)$$

This theoretical result implies that the cross-lingual similarity function derived from embeddings based on sentence IDs is essentially a generalization of the Dice coefficient.

5.3 SGNS with Sentence IDs

The Dice coefficient appears to be a particularly naïve variant of matrix-based methods that use sentence IDs. For example, Inverted Index (Søgaard et al., 2015)), which uses SVD over IDF followed by L_2 normalization (instead of L_1 normalization), shows significantly better performance. We propose using a third variant, sentence-ID SGNS (SID-SGNS), which simply applies SGNS (Mikolov et al., 2013b) to the word/sentence-ID matrix (see §5.1).

Table 3 compares its performance (Bilingual SID-SGNS) to the other methods, and shows that indeed, this algorithm behaves similarly to other sentence-ID-based methods. We observe similar results for other variants as well, such as SVD over positive PMI (not shown).

5.4 Embedding Multiple Languages

Up until now, we used bilingual data to train cross-lingual embeddings, even though our parallel corpus (the Bible) is in fact multi-lingual. Can we make better use of this fact?

An elegant property of the sentence-ID feature set is that it is a truly inter-lingual representation. This means that multiple languages can be represented together in the same matrix before factorizing it. This raises a question: does dimensionality reduction over a *multi*-lingual matrix produce better cross-lingual vectors than doing so over a bilingual matrix?

We test our hypothesis by comparing the performance of embeddings trained with SID-SGNS over all 57 languages of the Bible corpus to that of the bilingual embeddings we used earlier. This consistently improves performance across all the development benchmarks, providing a 4.69% average increase in performance (Table 3). With this

			Prior Art			This Work	
			Bilingual Autoencoders	Inverted Index	IBM Model-1	Bilingual SID-SGNS	Multilingual SID-SGNS
Word Alignment (1-AER)	GRAÇA	en fr	.4376	.3499	.4263	.4167	.4433
		fr en	.4488	.3995	.4248	.4300	.4632
		en es	.5000	.3443	.4251	.4200	.4893
		es en	.5076	.4545	.4243	.3610	.5015
		en pt	.4449	.3263	.4729	.3983	.4047
		pt en	.4474	.3902	.4712	.4272	.4151
	HANSARDS	en fr	.4083	.3336	.4360	.3810	.4091
		fr en	.4218	.3749	.4499	.3806	.4302
	LAMBERT	en es	.2960	.2268	.2400	.2471	.2989
		es en	.2905	.2696	.2443	.2415	.3049
	MIHALCEA	en ro	.2366	.1951	.2335	.1986	.2514
		ro en	.2545	.2133	.2214	.1914	.2753
	HOLMQVIST	en sv	.2746	.2357	.3405	.2373	.2737
		sv en	.2994	.2881	.3559	.2853	.3195
CAKMAK	en tr	.2256	.1731	.3154	.1547	.2404	
	tr en	.2661	.2665	.3494	.1571	.2945	
Dictionary Induction (P@1)	WIKTIONARY	en fr	.2475	.3125	.1791	.3182	.3304
		fr en	.2762	.3466	.1816	.3379	.3893
		en es	.2738	.3135	.0903	.3268	.3509
		es en	.3012	.3574	.1131	.3483	.3868
		en pt	.3281	.3866	.3779	.3869	.4058
		pt en	.3661	.4190	.4358	.4119	.4376
		en ar	.0995	.1364	.1316	.1364	.1605
		ar en	.1958	.2825	.2873	.2408	.3082
		en fi	.0887	.1367	.1340	.1280	.1591
		fi en	.1597	.2477	.2394	.1877	.2584
		en he	.0985	.1284	.1224	.1403	.1448
		he en	.1701	.2179	.2000	.1791	.2403
		en hu	.1679	.2182	.2219	.2299	.2482
		hu en	.2234	.3204	.2985	.2759	.3372
en tr	.1770	.2245	.1985	.2207	.2437		
tr en	.2069	.2835	.3073	.2598	.3080		
Average			0.2856	0.2867	0.2922	0.2830	0.3289
Top 1			2	0	8	0	22

Table 3: The performance of SID-SGNS compared to state-of-the-art cross-lingual embedding methods and traditional alignment methods.

advantage, SID-SGNS performs significantly better than the other methods combined.⁶ This result is similar in vein to recent findings in the parsing literature (Ammar et al., 2016; Guo et al., 2016), where multi-lingual transfer was shown to improve upon bilingual transfer.

In absolute terms, Multilingual SID-SGNS’s performance is still very low. However, this experiment demonstrates that one way of making significant improvement in cross-lingual embeddings is by considering additional sources of information, such as the multi-lingual signal demonstrated here. We hypothesize that, regardless of the algorithmic approach, relying solely on sentence IDs

⁶We observed a similar increase in performance when applying the multi-lingual signal to Søgaard et al.’s (2015) IDF-based method and to SVD over positive PMI.

from bilingual parallel corpora will probably not be able to improve much beyond IBM Model-1.

6 Data Paradigms

In §2, we assumed that using sentence-aligned data is a better approach than utilizing document-aligned data. Is this the case?

To compare the data paradigms, we run the same algorithm, SID-SGNS, also on document IDs from Wikipedia.⁷ We use the bilingual (not multilingual) version for both data types to control for external effects. During evaluation, we use a common vocabulary for both sentence-aligned and document-aligned embeddings.

⁷We use the word-document matrix mined by Søgaard et al. (2015), which contains only a subset of our target languages: English, French, and Spanish.

		The Bible	Wikipedia
GRAÇA	en fr	.3169	.2602
	fr en	.3089	.2440
	en es	.3225	.2429
	es en	.3207	.2504
HANSARDS	en fr	.3661	.2365
	fr en	.3345	.1723
LAMBERT	en es	.2161	.1215
	es en	.2123	.1027
WIKTIONARY	en fr	.3232	.3889
	fr en	.3418	.4135
	en es	.3307	.3262
	es en	.3509	.3310
Average		.3121	.2575
Top 1		10	2

Table 4: The performance of SID-SGNS with sentence-aligned data from the Bible (31,102 verses) vs document-aligned data from Wikipedia (195,000 documents).

Table 4 shows that using sentence IDs from the Bible usually outperforms Wikipedia. This remarkable result, where a small amount of parallel sentences is enough to outperform one of the largest collections of multi-lingual texts in existence, indicates that document-aligned data is an inferior paradigm for translation-related tasks such as word alignment and dictionary induction.

7 Conclusions

In this paper, we draw both empirical and theoretical parallels between modern cross-lingual word embeddings based on sentence alignments and traditional word alignment algorithms. We show the importance of sentence ID features and present a new, strong baseline for cross-lingual word embeddings, inspired by the Dice aligner. Our results suggest that apart from faster algorithms and more compact representations, recent cross-lingual word embedding algorithms are still unable to outperform the traditional methods by a significant margin. However, introducing our new multi-lingual signal considerably improves performance. Therefore, we hypothesize that the information in bilingual sentence-aligned data has been thoroughly mined by existing methods, and suggest that future work explore additional sources of information in order to make substantial progress.

Acknowledgments

The work was supported in part by The European Research Council (grant number 313695) and The Israeli Science Foundation (grant number

1555/15). We would like to thank Sarath Chandar for helping us run Bilingual Autoencoders on large datasets.

References

- Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.
- Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- Talha Cakmak, Süleyman Acar, and Gülsen Eyrgit. 2012. Word alignment for english-turkish language pair. In *LREC*.
- Sarath A P Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc.
- Christos Christodoulopoulos and Mark Steedman. 2015. A massively parallel corpus: The bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.
- Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics.
- Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, Denver, Colorado, May–June. Association for Computational Linguistics.
- Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 748–756.
- Joao Graca, Joana Pardal, Luísa Coheur, and Diamantino Caseiro. 2008. Building a golden collection of parallel multi-language word alignments. In *LREC*.
- Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In

- Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2734–2740. AAAI Press.
- Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Distributed Representations without Word Alignment. In *Proceedings of ICLR*, April.
- Maria Holmqvist and Lars Ahrenberg. 2011. A gold standard for englishswedish word alignment. In *NODALIDA*.
- Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India, December. The COLING 2012 Organizing Committee.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, volume 5, pages 79–86.
- Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.
- Patrik Lambert, Adria de Gispert, Rafael Banchs, and Jose Marino. 2005. Guidelines for word alignment evaluation and manual alignment. *Language Resources and Evaluation*, 39(4):267–285.
- Omer Levy and Yoav Goldberg. 2014. Neural word embeddings as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185.
- Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.
- Minh-Thang Luong, Hieu Pham, and Christopher Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 151–159, Denver, Colorado, May–June. Association for Computational Linguistics.
- Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*.
- Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.
- Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
- Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.
- Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1713–1722, Beijing, China, July. Association for Computational Linguistics.
- Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.
- Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1670, Berlin, Germany, August. Association for Computational Linguistics.
- Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.
- Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA, October. Association for Computational Linguistics.

Online Learning of Task-specific Word Representations with a Joint Biconvex Passive-Aggressive Algorithm

Pascal Denis¹ and Liva Ralaivola²

¹MAGNET, Inria Lille – Nord Europe, Villeneuve d’Ascq, France

pascal.denis@inria.fr

²QARMA, LIF, Aix-Marseille University, Marseille, France

liva.ralaivola@lif.univ-mrs.fr

Abstract

This paper presents a new, efficient method for learning task-specific word vectors using a variant of the Passive-Aggressive algorithm. Specifically, this algorithm learns a word embedding matrix in tandem with the classifier parameters in an online fashion, solving a biconvex constrained optimization at each iteration. We provide a theoretical analysis of this new algorithm in terms of regret bounds, and evaluate it on both synthetic data and NLP classification problems, including text classification and sentiment analysis. In the latter case, we compare various pre-trained word vectors to initialize our word embedding matrix, and show that the matrix learned by our algorithm vastly outperforms the initial matrix, with performance results comparable or above the state-of-the-art on these tasks.

1 Introduction

Recently, distributed word representations have become a crucial component of many natural language processing systems (Koo et al., 2008; Turian et al., 2010; Collobert et al., 2011). The main appeal of these word embeddings is twofold: they can be derived directly from raw text data in an unsupervised or weakly-supervised manner, and their latent dimensions condense interesting distributional information about the words, thus allowing for better generalization while also mitigating the presence of rare and unseen terms. While there are now many different spectral, probabilistic, and deep neural approaches for building vectorial word representations, there is still no clear understanding as to which syntactic and semantic information they really cap-

ture and whether or how these representations really differ (Chen et al., 2013; Levy and Goldberg, 2014b; Schnabel et al., 2015). Also poorly understood is the relation between the word representations and the particular learning algorithm (e.g., whether linear or non-linear) that uses them as input (Wang and Manning, 2013).

What seems clear, however, is that there is no single best embedding and that their impact is very much task-dependent. This in turn raises the question of how to learn word representations that are adapted to a particular task and learning objective. Three different research routes have been explored towards learning task-specific word embeddings. A first approach (Collobert et al., 2011; Maas et al., 2011) is to learn the embeddings for the target problem jointly with additional unlabeled or (weakly-)labeled data in a semi-supervised or multi-task approach. While very effective, this joint training typically requires large amounts of data and often prohibitive processing times in the case of multi-layer neural networks (not to mention their lack of theoretical learning guarantees in part due to their strong non-convexity). Another approach consists in training word vectors using some existing algorithm as in like word2vec (Mikolov et al., 2013) in a way that exploits prior domain knowledge (e.g., by defining more informative, task-specific contexts) (Bansal et al., 2014; Levy and Goldberg, 2014a). In this case, there is still a need for additional weakly- or hand-labeled data, and there is no guarantee that the newly learned embeddings will indeed benefit the performance, as they are trained independently of the task objective. A third approach is to start with some existing pre-trained embeddings and fine-tune them to the task by integrating them in a joint learning objective either using backpropagation (Lebret et al., 2013) or regularized logistic regression (Labutov and Lipson, 2013). These

approaches in effect hit a sweet spot by leveraging pre-trained embeddings and requiring no additional data or domain knowledge, while directly tying them to task learning objective.

Inspired by these latter approaches, we propose a new, online soft-margin classification algorithm, called Re-embedding Passive-Aggressive (or RPA), that jointly learns an embedding matrix in tandem with the model parameters. As its name suggests, this algorithm generalizes the Passive-Aggressive (PA) algorithm of Crammer and Singer (2006) by allowing the data samples to be projected into the lower dimension space defined by the original embedding matrix. Our approach may be seen as extending the work of (Grandvalet and Canu, 2003) which addresses the problem of simultaneously learning the features and the weight vector of an SVM classifier. An important departure, beyond the online nature of RPA, is that it learns a projection matrix and not just a diagonal one (which is essentially what this earlier work does). Our approach and analysis are also related to (Blondel et al., 2014), which tackles non-negative matrix factorization with the PA philosophy.

The main contributions of this paper are as follows. First, we derive a new variant of the Passive-Aggressive algorithm able to jointly learn an embedding matrix along with the weight vector of the model (section 2). Second, we provide theoretical insights as to bound the cumulative squared loss of our learning procedure over any given sequence of examples—the results we give are actually related to a learning procedure that slightly differs from the algorithm we introduce but that is more easily and compactly amenable to a theoretical study. Third, we further study the behavior of this algorithm on synthetic data (section 4) and we finally show that it performs well on five real-world NLP classification problems (section 5).

2 Algorithm

We consider the problem of learning a binary linear classification function $f_{\Phi, w}$, parametrized by both a weight vector $w \in \mathbb{R}^k$ and an embedding matrix $\Phi \in \mathbb{R}^{k \times p}$ (typically, with $k \ll p$), which is defined as:

$$f_{\Phi, w} : \mathcal{X} \subset \mathbb{R}^p \rightarrow \{-1, +1\}$$

$$x \mapsto \text{sign}(\langle w, \Phi x \rangle)$$

We aim at an online learning scenario, wherein both w and Φ will be updated in a sequential fashion. Given a labeled data steam $S =$

$\{(\mathbf{x}_t, y_t)\}_{t=1}^T$, it seems relevant at each step to solve the following soft-margin constrained optimization problem:

$$\underset{\substack{w \in \mathbb{R}^k \\ \Phi \in \mathbb{R}^{k \times p}}}{\text{argmin}} \frac{1}{2} \|w - w_t\|_2^2 + \frac{\lambda}{2} \|\Phi - \Phi_t\|_F^2 + C\xi^2 \quad (1a)$$

$$\text{s.t. } \ell_t(w; \Phi; \mathbf{x}_t) \leq \xi \quad (1b)$$

where $\|\cdot\|_2$, $\|\cdot\|_F$ stand for the l_2 and Frobenius norms, respectively, and C controls the “aggressiveness” of the update (as larger C values imply updates that are directly proportional to the incurred loss). We define $\ell_t(w; \Phi; \mathbf{x}_t)$ as the hinge loss, that is:

$$\ell_t(w; \Phi; \mathbf{x}_t) \doteq \max(0, 1 - y_t \langle w, \Phi \mathbf{x}_t \rangle). \quad (2)$$

The optimization problem in (1) is reminiscent of the soft-margin Passive-Aggressive algorithm proposed in (Crammer et al., 2006) (specifically, PA-II), but both the objective and the constraint now include a term based on the embedding matrix Φ . The λ regularization parameter in the objective controls the allowed divergence in the embedding parameters between iterations.

Interestingly, the new objective remains convex, but the margin constraint doesn’t as it involves a multiplicative term between the weight vector and the embedding matrix, making the overall problem bi-convex (Gorski et al., 2007). That is, the problem is convex in w for fixed values of Φ and convex in Φ for fixed values of w . Incidentally, the formulation presented by (Labutov and Lipson, 2013) is also bi-convex (as it also involves a similar multiplicative term), although the authors proceed as if it were jointly convex (i.e., convex in both w and Φ). In order to solve this problem, we resort to an alternating update procedure which updates each set of parameters (i.e., either the weight vector or the embedding matrix) while holding the other fixed until some stopping criterion is met (in our case, the value of the objective doesn’t change). As shown in Algorithm 1, this procedure allows us to compute closed-form updates similar to those of PA, and to make use of the same theoretical apparatus for analyzing RPA.

2.1 Unified Formalization

Suppose from now on that C and λ are fixed and so are w_t and Φ_t : this will allow us to drop the explicit dependence on these values and to keep

Algorithm 1 Re-embedding Passive-Aggressive

Require:

- $S = \{(\mathbf{x}_t, y_t)\}_{t=1}^T$, a stream of data
- a base embedding Φ_0

Ensure:

- a classification vector \mathbf{w}
- a re-embedding matrix Φ

Initialize \mathbf{w}_0 $t \leftarrow 0$ **repeat**

$$\mathbf{w}_{t+1}^0 \leftarrow \mathbf{w}_t, \Phi_{t+1}^0 \leftarrow \Phi_t,$$

$$n \leftarrow 1$$

repeat

$$\mathbf{w}_{t+1}^{n+1} \leftarrow \mathbf{w}_{t+1}^n + \frac{\ell_t(\mathbf{w}_{t+1}^n; \Phi_{t+1}^n)}{\|\Phi_{t+1}^n \mathbf{x}_t\|_2^2 + \frac{\lambda}{2C}} y_t \mathbf{x}_t$$

$$\Phi_{t+1}^{n+1} \leftarrow \Phi_{t+1}^n + \frac{\ell_t(\mathbf{w}_{t+1}^{n+1}; \Phi_{t+1}^n)}{\|\mathbf{w}_{t+1}^{n+1}\|_2^2 \|\mathbf{x}_t\|_2^2 + \frac{\lambda}{2C}} y_t \mathbf{w}_{t+1}^{n+1} \mathbf{x}_t^\top.$$

with

$$\ell_t(\mathbf{w}; \Phi) = \max(0, 1 - y_t \langle \mathbf{w}, \Phi \mathbf{x}_t \rangle).$$

until $\mathbf{w}_{t+1}^n \rightarrow \mathbf{w}_{t+1}^\infty$ and $\Phi_{t+1}^n \rightarrow \Phi_{t+1}^\infty$
Update \mathbf{w}_{t+1} and Φ_{t+1} :

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1}^\infty \quad \Phi_{t+1} \leftarrow \Phi_{t+1}^\infty$$

until some stopping criterion is met**return** \mathbf{w}_t, Φ_t

the notation light. Let Q_t be defined as:

$$Q_t(\mathbf{w}, \Phi, \xi) \doteq \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 + \frac{\lambda}{2} \|\Phi - \Phi_t\|_F^2 + C\xi^2. \quad (3)$$

and margin q_t be defined as:

$$q_t(\mathbf{w}, \Phi) \doteq 1 - \langle \mathbf{w}, y_t \Phi \mathbf{x}_t \rangle \quad (4a)$$

$$= 1 - \left\langle \Phi, y_t \mathbf{w} \mathbf{x}_t^\top \right\rangle_F. \quad (4b)$$

Here $\langle \cdot, \cdot \rangle_F$ denote the Frobenius inner product. We have purposely provided the two equivalent forms (4a) and (4b) to emphasize the (syntactic) exchangeability of \mathbf{w} and Φ . As we shall see, this is going to be essential to derive an alternating update procedure—which alternates the updates with respect to \mathbf{w} and Φ —in a compact way.

Given Q_t and q_t , we are now interested in solving the following optimization problem:

$$\mathbf{w}_{t+1}, \Phi_{t+1}, \xi_{t+1} = \operatorname{argmin}_{\mathbf{w}, \Phi, \xi} Q_t(\mathbf{w}, \Phi, \xi) \quad (5a)$$

$$\text{s.t. } q_t(\mathbf{w}, \Phi) \leq \xi. \quad (5b)$$

2.2 Bi-convexity

It turns out that problem (5) is a bi-convex optimization problem: it is indeed straightforward to observe that if Φ is fixed then the problem is convex in (\mathbf{w}, ξ) —it is the classical passive-aggressive II optimization problem—and if \mathbf{w} is fixed then the problem is convex in (Φ, ξ) . If there exist theoretical results on the solving of bi-convex optimization problems, the machinery to use pertains to combinatorial optimization, which might be too expensive. In addition, computing the solution of (5) would drive us away from the spirit of passive-aggressive learning which rely on cheap and statistically meaningful (from the mistake-bound perspective) updates. This is the reason why we propose to resort to an alternate online procedure to solve a proxy of (5).

2.3 An Alternating Online Procedure

Instead of tackling problem (5) directly we propose to solve, at each time t , either of:

$$\mathbf{w}_{t+1}, \xi_{t+1} = \operatorname{argmin}_{\mathbf{w}, \xi} Q_t(\mathbf{w}, \Phi_t, \xi) \text{ s.t. } q_t(\mathbf{w}, \Phi_t) \leq \xi, \quad (6)$$

$$\Phi_{t+1}, \xi_{t+1} = \operatorname{argmin}_{\Phi, \xi} Q_t(\mathbf{w}_t, \Phi, \xi) \text{ s.t. } q_t(\mathbf{w}_t, \Phi) \leq \xi. \quad (7)$$

This means that the optimization is performed with either Φ fixed to Φ_t or \mathbf{w} fixed to \mathbf{w}_t .

Informally, the iterative Algorithm 1, resulting from this alternate scheme, will solve at each round a simple constrained optimization problem, in which the objective is to minimize the squared Euclidean distances between the new weight vector (resp. the new embedding matrix) and the current one, while making sure that both sets of parameters achieve a correct prediction with a sufficiently high margin. Note that one may recover the standard passive-aggressive algorithm by simply fixing the embedding matrix to the identity matrix. Also note that if the right stopping criteria are retained, Algorithm 1 is guaranteed to converge to a local optimum of (5) (see (Gorski et al., 2007)).

When fully developed, problems (6)-(7) respectively write as:

$$\mathbf{w}_{t+1}, \xi_{t+1} = \operatorname{argmin}_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 + C\xi^2$$

$$\text{s.t. } 1 - \langle \mathbf{w}, y_t \Phi_t \mathbf{x}_t \rangle \leq \xi,$$

or

$$\begin{aligned} \Phi_{t+1}, \xi_{t+1} = \operatorname{argmin}_{\Phi, \xi} & \frac{\lambda}{2} \|\Phi - \Phi_t\|_F^2 + C\xi^2 \\ \text{s.t. } & 1 - \langle \Phi, y_t \mathbf{w}_t \mathbf{x}_t^\top \rangle_F \leq \xi. \end{aligned}$$

Using the equivalence in (4), both problems can be seen as special instances of the generic problem:

$$\mathbf{u}^*, \xi^* = \operatorname{argmin}_{\mathbf{u}, \xi} \frac{\lambda}{2} \|\mathbf{u} - \mathbf{u}_t\|_*^2 + C\xi^2 \quad (8a)$$

$$\text{s.t. } 1 - \langle \mathbf{u}, \mathbf{v}_t \rangle_* \leq \xi. \quad (8b)$$

where $\|\cdot\|_*$ and $\langle \cdot, \cdot \rangle_*$ are generalized versions of the l_2 norm and the inner product, respectively.

This is a convex optimization problem that can be readily solved using classical tools from convex optimization to give:

$$\mathbf{u}^* = \mathbf{u}_t + \tau_u \mathbf{v}_t, \text{ with } \tau_u = \frac{\max(0, 1 - \langle \mathbf{u}_t, \mathbf{v}_t \rangle_*)}{\|\mathbf{v}_t\|_* + \frac{\lambda}{2C}}, \quad (9)$$

which comes from the following proposition.

Proposition 1. *The solution of (8) is (9).*

Proof. The Lagrangian associated with the problem is given by:

$$\mathcal{L}(\mathbf{u}, \xi, \tau) = \frac{\lambda}{2} \|\mathbf{u} - \mathbf{u}_t\|_*^2 + C\xi^2 + \tau (1 - \xi - \langle \mathbf{u}, \mathbf{v}_t \rangle_*) \quad (10)$$

with $\tau > 0$.

Necessary conditions for optimality are $\nabla_{\mathbf{u}} \mathcal{L} = \mathbf{0}$, and $\nabla_{\xi} \mathcal{L} = 0$, which imply $\mathbf{u} = \mathbf{u}_t + \frac{\tau}{\lambda} \mathbf{v}_t$, and $\xi = \frac{\tau}{2C}$. Using this in (10) gives the function:

$$g(\tau) = \frac{\tau^2 \|\mathbf{v}_t\|_*^2}{2\lambda} + \frac{\tau^2}{4C} + \tau - \frac{\tau^2}{2C} - \tau \langle \mathbf{u}_t, \mathbf{v}_t \rangle_* - \frac{\tau^2 \|\mathbf{v}_t\|_*^2}{\lambda},$$

which is maximized with respect to τ when $g'(\tau) = 0$, i.e.:

$$\left(\frac{\|\mathbf{v}_t\|_*^2}{\lambda} - \frac{1}{2C} - \frac{2\|\mathbf{v}_t\|_*^2}{\lambda} \right) \tau + 1 - \langle \mathbf{u}_t, \mathbf{v}_t \rangle_* = 0.$$

Taking into account the constraint $\tau \geq 0$, the maximum of g is attained at $\tilde{\tau}$ with:

$$\tilde{\tau} = \frac{\lambda \max(0, 1 - \langle \mathbf{u}_t, \mathbf{v}_t \rangle_*)}{\|\mathbf{v}_t\|_*^2 + \frac{\lambda}{2C}},$$

which, setting $\tau_u = \tilde{\tau}/\lambda$ gives (9). \square

The previous proposition allows us to readily have the solutions of (6) and (7) as follows.

Proposition 2. *The updates induced by the solutions of (6) and (7) are respectively given by:*

$$\mathbf{w}^* = \mathbf{w}_t + \tau_w y_t \mathbf{x}_t, \text{ with } \tau_w = \frac{\ell_t(\mathbf{w}_t; \Phi_t)}{\|\Phi_t \mathbf{x}_t\|_2^2 + \frac{1}{2C}} \quad (11)$$

$$\Phi^* = \Phi_t + \tau_\Phi y_t \mathbf{w}_t \mathbf{x}_t^\top, \text{ with } \tau_\Phi = \frac{\ell_t(\mathbf{w}_t; \Phi_t)}{\|\mathbf{w}_t\|_2^2 \|\mathbf{x}_t\|_2^2 + \frac{\lambda}{2C}}, \quad (12)$$

where $\ell_t(\mathbf{w}; \Phi) = \max(0, q_t(\mathbf{w}, \Phi))$.

Proof. Just instantiate the results of Proposition 1 with $(\mathbf{u}, \mathbf{v}_t) = (\mathbf{w}, y_t \Phi_t \mathbf{x}_t)$ for (11) and $(\mathbf{u}, \mathbf{v}_t) = (\Phi, y_t \mathbf{x}_t \mathbf{w}_t^\top)$ for (12). \square

Remark 1 (Hard-margin case). *Note that the hard-margin version of the previous problem:*

$$\operatorname{argmin}_{\substack{\mathbf{w} \in \mathbb{R}^k \\ \Phi \in \mathbb{R}^{k \times p}}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 + \frac{\lambda}{2} \|\Phi - \Phi_t\|_F^2 \quad (13)$$

$$\text{s.t. } 1 - y_t \langle \mathbf{w}, \Phi \mathbf{x}_t \rangle \leq 0 \quad (14)$$

is degenerate from the alternated optimization point of view. It suffices to observe that the updates entailed by the hard-margin problem correspond to (9) with C set to ∞ ; if it happens that either $\Phi_0 = \mathbf{0}$ or $\mathbf{w}_0 = \mathbf{0}$, then one of the optimization problems (wrt to \mathbf{w} or Φ) has no solution.

3 Analysis

Using the same technical tools as in (Crammer et al., 2006), and the unified formalization of section 2, we have the following result.

Proposition 3. *Suppose that problem (8) is iteratively solved for a sequence of vectors $\mathbf{v}_1, \dots, \mathbf{v}_T$ to give $\mathbf{u}_2, \dots, \mathbf{u}_{T+1}$, \mathbf{u}_1 being given. Suppose that, at each time step, $\|\mathbf{v}_t\|_* \leq R$, for some $R > 0$. Let \mathbf{u}^* be an arbitrary vector living in the same space as \mathbf{u}_1 . The following result holds*

$$\sum_{t=1}^T \ell_t^2 \leq \left(R^2 + \frac{\lambda}{2C} \right) \left(\|\mathbf{u}^* - \mathbf{u}_1\|_*^2 + \frac{2C}{\lambda} \sum_{t=1}^T (\ell_t^*)^2 \right) \quad (15)$$

where

$$\ell_t = \max(0, 1 - \langle \mathbf{u}_t, \mathbf{v}_t \rangle_*), \text{ and } \ell_t^* = \max(0, 1 - \langle \mathbf{u}^*, \mathbf{v}_t \rangle_*). \quad (16)$$

This proposition and the accompanying lemmas are simply a variation of the results of (Crammer et al., 2006), with the addition that they are based on the generic problem (8). The loss bound applies for a version of Algorithm 1 where one of the parameters, either the weight vector or the re-embedding matrix, is kept fixed for some time.

Proposition 3 makes use of the following lemma (see Lemma 1 in (Crammer et al., 2006)):

Lemma 1. Suppose that problem (8) is iteratively solved for a sequence of vectors $\mathbf{v}_1, \dots, \mathbf{v}_T$ to give $\mathbf{u}_2, \dots, \mathbf{u}_{T+1}$, \mathbf{u}_1 being given. The following holds:

$$\sum_{t=1}^T \tau_u (2\ell_t - \tau_u \|\mathbf{v}_t\|_*^2 - 2\ell_t^*) \leq \|\mathbf{u}^* - \mathbf{u}_1\|_*^2. \quad (17)$$

Proof. As in (Crammer et al., 2006), simply set $\Delta_t = \|\mathbf{u}_t - \mathbf{u}^*\|_*^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_*^2$ and bound $\sum_{\tau=1}^T \Delta_t$ from above and below. For the upper bound: $\sum_{\tau=1}^T \Delta_t = \|\mathbf{u}_1 - \mathbf{u}^*\|_*^2 - \|\mathbf{u}_{T+1} - \mathbf{u}^*\|_*^2 \leq \|\mathbf{u}_1 - \mathbf{u}^*\|_*^2$.

For the lower bound, we focus on the nontrivial situation where $\ell_t > 0$, otherwise $\Delta_t = 0$ (i.e. no update is made) and the bounding is straightforward. Making use of the value of τ_u :

$$\begin{aligned} \Delta_t &= \|\mathbf{u}_t - \mathbf{u}^*\|_*^2 - \|\mathbf{u}_{t+1} - \mathbf{u}^*\|_*^2 \\ &= \|\mathbf{u}_t - \mathbf{u}^*\|_*^2 - \|\mathbf{u}_t + \tau_u \mathbf{v}_t - \mathbf{u}^*\|_*^2 \quad (\text{see (8)}) \\ &= -2\tau_u \langle \mathbf{u}_t - \mathbf{u}^*, \mathbf{v}_t \rangle_* - \tau_u^2 \|\mathbf{v}_t\|_*^2 \\ &= -2\tau_u \langle \mathbf{u}_t, \mathbf{v}_t \rangle_* + 2\tau_u \langle \mathbf{u}^*, \mathbf{v}_t \rangle_* - \tau_u^2 \|\mathbf{v}_t\|_*^2. \end{aligned}$$

Since $\ell_t > 0$, then $\langle \mathbf{u}_t, \mathbf{v}_t \rangle_* = 1 - \ell_t$; also, by the definition of ℓ_t^* (see (16)), $\ell_t^* \geq 1 - \langle \mathbf{u}^*, \mathbf{v}_t \rangle_*$, or $\langle \mathbf{u}^*, \mathbf{v}_t \rangle_* \geq 1 - \ell_t^*$. This readily gives

$$\begin{aligned} \Delta_t &\geq -2\tau_u(1 - \ell_t) + 2\tau_u(1 - \ell_t^*) - \tau_u^2 \|\mathbf{v}_t\|_*^2 \\ &= 2\tau_u \ell_t - 2\tau_u \ell_t^* - \tau_u^2 \|\mathbf{v}_t\|_*^2, \end{aligned}$$

hence the targeted lower bound and, in turn, (17). \square

Proof of Proposition 3. As for the proof of Proposition 3, it suffices, again, to follow the steps given in (Crammer et al., 2006), but this time, for the proof of Theorem 5.

Note that for any $\beta \neq 0$, $(\beta\tau_u - \ell_t^*/\beta)^2$ is well-defined and nonnegative and thus:

$$\begin{aligned} &\|\mathbf{u}^* - \mathbf{u}_1\|_*^2 \\ &\geq \sum_{t=1}^T [\tau_u (2\ell_t - \tau_u \|\mathbf{v}_t\|_*^2 - 2\ell_t^*) - (\beta\tau_u - \ell_t^*/\beta)^2] \\ &= \sum_{t=1}^T (2\tau_u \ell_t - \tau_u^2 (\|\mathbf{v}_t\|_*^2 + \beta^2) - 2\tau_u \ell_t^* \\ &\quad + 2\tau_u \ell_t^* - (\ell_t^*)^2 / \beta^2) \\ &= \sum_{t=1}^T (2\tau_u \ell_t - \tau_u^2 (\|\mathbf{v}_t\|_*^2 + \beta^2) - (\ell_t^*)^2 / \beta^2). \end{aligned}$$

Setting $\beta = \sqrt{\lambda/2C}$ and using $\tau_u = \ell_t / (\|\mathbf{v}\|_*^2 +$

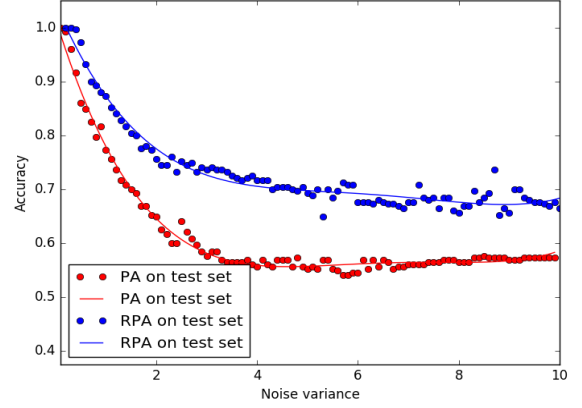


Figure 1: Accuracy rates for PA and RPA as a function of the variance of the Gaussian noise added to the “true” embedding Φ . The observed X matrix is $n = 500 \times p = 1000$.

$\frac{\lambda}{2C}$) (see (9)) gives

$$\begin{aligned} &\|\mathbf{u}^* - \mathbf{u}_1\|_*^2 \\ &\geq \sum_{t=1}^T \left(2\tau_u \ell_t - \tau_u^2 \left(\|\mathbf{v}_t\|_*^2 + \frac{\lambda}{2C} \right) - \frac{2C}{\lambda} (\ell_t^*)^2 \right), \\ &= \sum_{t=1}^T \left(\frac{\ell_t^2}{\|\mathbf{v}_t\|_*^2 + \frac{\lambda}{2C}} - \frac{2C}{\lambda} (\ell_t^*)^2 \right). \end{aligned}$$

Using the assumption that $\|\mathbf{v}_t\|_* \leq R$ for all t and rearranging terms concludes the proof. \square

The result given in Proposition 3 bounds the cumulative loss of the learning procedure when one of the parameters, either Φ or w , is fixed and the other is the optimization variable. Therefore, it does not directly capture the behavior of Algorithm 1, which alternates between the updates of Φ and w . A proper analysis of Algorithm 1 would require a refinement of Lemma 1 which, to our understanding, would be the core of a new result. This is a problem we intend to put our energy on in the near future, as an extension to this work.

4 Experiments on Synthetic Data

In order to better understand and validate the RPA algorithm, we first conducted some synthetic experiments. Specifically, we simulated a high-dimensional matrix $X \in \mathbb{R}^{n \times p}$, with n data samples realizing p “words” and $p \gg n$, using the following generative model:

$$X = Z\tilde{\Phi}, \text{ where } \tilde{\Phi} = \Phi + \mathcal{E}$$

That is, each p -dimensional data point x_i was generated from a hidden lower k -dimensional z_i and

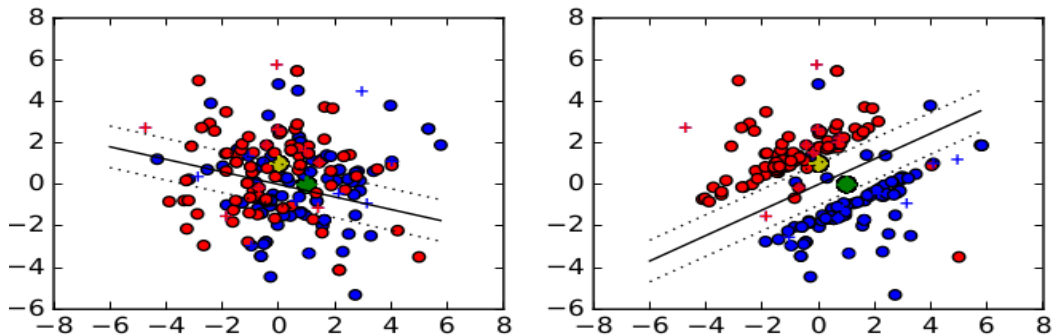


Figure 2: Hyper-plane learned by PA on $X\tilde{\Phi}^\top$ (left pane) compared to hyper-plane and data representations learned by RPA (right pane). X is $n = 200 \times p = 500$, and noise $\sigma = 2$. Training and test points appear as circles or plus marks, respectively. RPA’s hyper-parameters are set to $C = 100$ and $\lambda = .5$.

an embedding matrix $\Phi \in \mathbb{R}^{k \times p}$, mapping k latent concepts to the p observed words. For simplicity, we assume that: (i) there are only two concepts (i.e., $k = 2$), (ii) each data point realizes a single concept (i.e., each x_i is a p -dimensional indicator vector), (iii) each concept is equally represented in the data (with $n/2$ data points), and (iv) each concept deterministically signals a class label, either -1 or $+1$. Recovering Z and predicting the z_i ’s labels is trivial if one is given X and the true embedding Φ , so we added Gaussian noise $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_p)$ to each ϕ_i . The resulting, observed noisy matrix is denoted $\tilde{\Phi}$.

Given this setting, the goal of the RPA is to learn a set of classification parameters $w \in \mathbb{R}^k$ in the latent space and to “de-noise” the observed embedding matrix $\tilde{\Phi}$ by exploiting the labeled data. We are interested in comparing the RPA with a regular PA that directly learns from the noisy data $X\tilde{\Phi}^\top$. The outcome of this comparison is plotted in Figure 1. For this experiment, we randomly splitted the X data according to 80/10/10 for train/dev/test and considered increasing noise variance from 0 to 10 by increment of 0.1. Each dot in Figure 1 corresponds to the average accuracy over 10 separate, random initializations of the embedding matrix at a particular noise level. Hyper-parameters were optimized using a grid search on the dev set for both the PA and RPA.¹

As shown in Figure 1, the PA’s accuracy quickly drops to levels that are only slightly above chance, while the RPA manages to maintain an accuracy close to .7 even with large noise. This indicates

that the RPA is able to recover some of the structure in the embedding matrix. This behavior is also illustrated in Figure 2, wherein the two hidden concepts appear in yellow and green. While the standard PA learns a very bad hyper-plane, which fails to separate the two concepts, the RPA learns a much better hyper-plane. Interestingly, most of the data points appear to have been projected on the margins of the hyper-plane.

5 Experiments on NLP tasks

This section assesses the effectiveness of RPA on several text classification tasks.

5.1 Evaluation Datasets

We consider five different classification tasks which are concisely summarized in Table 1.

20 Newsgroups Our first three text classification tasks from this dataset² consists in categorizing documents into two related sub-topics: (i) **Comp.**: IBM vs. Mac, (ii) **Religion**: atheism vs. christian, and (iii) **Sports**: baseball vs. hockey.

IMDB Movie Review This movie dataset³ was introduced by (Maas et al., 2011) for sentiment analysis, and contains 50,000 reviews, divided into a balanced set of highly positive (7 stars out of 10 or more) and negative scores (4 stars or less).

TREC Question Classification This dataset⁴ (Li and Roth, 2002) involves six question types: abbreviation, description, entity, human, location, and number.

¹We use $\{1, 5, 10, 50\}$ for the number iterations, C ’s values were $\{.1, .5, 1.0, 2.0, 10.0\}$, and λ was set to 10^k , with $k \in \{-2, -1, 0, 1, 2\}$.

²qwone.com/~jason/20Newsgroups

³ai.stanford.edu/~amaas/data/sentiment

⁴cogcomp.cs.illinois.edu/Data/QA/QC

Name	lab.	examples		Voc. size	
		Train	Test	All	$fr > 1$
comp	2	1 168	777	30 292	15 768
rel.	2	1 079	717	32 361	18 594
sports	2	1 197	796	33 932	19 812
IMDB	2	25k	25k	172 001	86 361
TREC	6	5 452	500	9 775	3 771

Table 1: Number of labels, samples, vocabulary sizes (incl. non-hapax words) per dataset.

5.2 Preprocessing and Document Vectors

All datasets were pre-processed with the Stanford tokenizer⁵, except for the TREC corpus which comes pre-tokenized. Case was left intact unless used in conjunction with word embeddings that assume down-casing (see below).

For constructing document or sentence vectors, we used a simple 0-1 bag-of-words model, simply summing over the word vectors of occurring tokens, followed by L2-normalization of the resulting vector in order to avoid document/sentence length effects. For each dataset, we restricted the vocabulary to non-hapax words (i.e., words occurring more than once). Words unknown to the embedding were mapped to zero vectors.

5.3 Initial Word Vector Representations

Five publicly available word vectors were used to define initial embedding matrices in the RPA. The coverage of the different embeddings wrt each dataset vocabulary is reported in Table 2.

CnW These word vectors were induced using the neural language model of (Collobert and Weston, 2008) re-implemented by (Turian et al., 2010).⁶ They were trained on 63M word news corpus, covering 268,810 word forms (intact case), with 50, 100 or 200 dimensions for each word.

HLBL These were obtained using the probabilistic Hierarchical log-bilinear language model of (Mnih and Hinton, 2009), again re-implemented by (Turian et al., 2010) with 50 or 100 dimensions. They cover 246,122 word forms.

HPCA (Lebret and Collobert, 2014) present a variant of Principal Component Analysis, Hellinger PCA, for learning spectral word vectors. These were trained over 1.62B words from

⁵nlp.stanford.edu/software/tokenizer.shtml

⁶metaoptimize.com/projects/wordreprs

		Word Embedding				
CnW	GV6	GV840	HPCA	HLBL	SkGr	
40.41	27.18	20.26	32.20	40.44	32.19	
25.45	13.07	8.95	16.90	25.39	17.33	
35.29	19.01	13.96	29.97	35.21	30.66	
39.79	12.67	4.93	22.15	39.63	16.79	
3.42	0.61	0.40	1.38	3.63	4.51	

Table 2: Out-of-vocabulary rates for non-hapax words in each dataset-embedding pair.

Wikipedia, RCV1, and WSJ with all words lower-cased, and digits mapped to a special symbol. Vocabulary is restricted to words occurring 100 times or more (hence, a total of 178,080 words). These come in 50, 100 and 200 dimensions.⁷

GloVe These global word vectors are trained using a log-bilinear regression model on aggregated global word co-occurrence statistics (Pennington et al., 2014). We use two different releases:⁸ (i) **GV6B** trained on Wikipedia 2014 and Gigaword 5 (amounting to 6B down-cased words and a vocabulary of 400k) with vectors in 50, 100, 200, or 300 dimensions, and (ii) **GV840B** trained over 840B uncased words (a 2.2M vocabulary) with vectors of length 300.

SkGr Finally, we use word vectors pre-trained with the skip-gram neural network model of (Mikolov et al., 2013): each word’s Huffman code is fed to a log-linear classifier with a continuous projection layer that predicts context words within a specified window. The embeddings were trained on a 100B word corpus of Google news data (a 3M vocabulary) and are of length 300.⁹

rand In addition to these pre-trained word representations, we also use random vectors of lengths 50, 100, and 200 as a baseline embedding. Specifically, each component of these word vector is uniformly distributed on the unit interval $(-1, 1)$.

5.4 Settings

The hyperparameters of the model, C , λ , and the number it of iterations over the training set, were estimated using a grid search over $C = 10^{k \in \{-6, -4, -2, 0, 2, 4, 6\}}$, $\lambda = 10^{l \in \{-3, -2, -1, 0, 1, 2, 3\}}$, and $it \in \{1, 5, 10\}$ in a 10-fold cross-validation

⁷lebret.ch/words

⁸nlp.stanford.edu/projects/glove

⁹code.google.com/archive/p/word2vec

Emb/Task Method	Size	comp		religion		sports		trec		imdb		Average	
		PA	RPA	PA	RPA	PA	RPA	PA	RPA	PA	RPA	PA	RPA
rand	50	54.57	87.13	60.67	91.91	63.07	92.34	56.80	88.40	61.47	86.47	59.32	89.25
	100	62.03	87.39	66.95	<u>92.19</u>	65.58	<u>93.22</u>	68.00	88.20	65.07	86.56	65.53	89.51
	200	65.51	<u>87.64</u>	73.22	<u>92.19</u>	71.11	92.96	70.00	88.20	69.10	<u>86.58</u>	69.79	89.51
CnW	50	50.32	85.97	55.09	91.91	56.91	93.72	69.40	87.20	58.68	86.54	58.08	89.07
	100	49.29	<u>87.13</u>	47.56	91.63	58.54	93.72	69.20	<u>87.80</u>	65.04	86.51	57.93	89.36
	200	50.32	87.00	53.14	91.77	63.94	93.72	75.80	87.60	68.21	<u>86.64</u>	62.28	89.35
HLBL	50	51.99	<u>87.13</u>	68.62	91.35	61.31	93.72	66.80	<u>88.80</u>	67.30	<u>86.70</u>	63.20	89.54
	100	53.54	86.74	65.83	<u>91.77</u>	63.07	93.72	75.60	88.20	72.41	86.68	66.09	89.42
HPCA	50	50.45	<u>86.87</u>	50.77	91.63	64.20	93.34	66.00	<u>88.60</u>	62.78	80.56	58.84	88.20
	100	50.45	<u>86.87</u>	47.98	91.63	64.57	<u>93.72</u>	72.00	88.40	64.25	85.33	59.85	89.19
	200	50.45	<u>86.87</u>	48.12	<u>91.91</u>	62.56	93.59	78.40	88.00	64.75	<u>85.84</u>	60.86	89.24
GV6B	50	55.21	86.87	75.59	91.91	86.68	95.23	65.80	<u>89.00</u>	75.12	86.41	71.68	89.88
	100	58.17	86.87	76.43	91.63	90.33	96.48	70.40	<u>89.00</u>	78.72	86.50	74.81	90.10
	200	70.40	86.87	73.22	91.63	93.34	97.11	76.60	<u>89.00</u>	81.55	<u>86.57</u>	79.02	90.24
	300	76.58	<u>87.13</u>	79.92	91.77	94.97	<u>97.74</u>	78.60	88.60	82.41	86.41	82.50	90.33
GV840B	300	75.80	<u>87.13</u>	88.15	<u>92.05</u>	88.69	<u>96.23</u>	77.20	<u>89.20</u>	84.17	<u>86.46</u>	82.80	90.21
SkGr	300	70.79	<u>87.39</u>	88.01	<u>92.05</u>	91.96	<u>97.61</u>	84.00	<u>90.60</u>	83.39	<u>88.52</u>	83.63	91.23
one-hot			<u>87.26</u>		<u>91.91</u>		<u>93.47</u>		<u>88.00</u>		<u>88.29</u>		89.786

Table 3: Accuracy results on our five datasets for the Re-embedding Passive-Aggressive (RPA) against standard PA-II (PA). For each task, the best results for each embedding method (across different dimensions) has been greyed out, while the overall highest accuracy score has been underlined.

over the training data. For the alternating on-line procedure, we used the difference between the objective values from one iteration to the next for defining the stopping criterion, with maximum number of iterations of 50. In practice, we found that the search often converged much few iterations. The multi-class classifier used for the TREC dataset was obtained by training the RPA in simple One-versus-All fashion, thus learning one embedding matrix per class.¹⁰ For datasets with label imbalance, we set a different C parameter for each class, re-weighting it in proportion to the inverse frequency of the class.

5.5 Results

Table 3 summarizes accuracy results for the RPA against those obtained by a PA trained with fixed pre-trained embeddings. The first thing to notice is that the RPA delivers massive accuracy improvements over the vanilla PA across datasets and embedding types and sizes, thus showing that the RPA is able to learn word representations that are better tailored to each problem. On average, accuracy gains are between 22% and 31% for CnW, HLBL, and HPCA. Sizable improvements, ranging from 8% and 18%, are also found for the better

¹⁰More interesting configurations (e.g., a single embedding matrix shared across classes), are left for future work.

performing GV6B, GV840B, and SkGr. Second, RPA is able to outperform on all five datasets the strong baseline provided by the one-hot version of PA trained on the original high-dimensional space, with some substantial gains on *sports* and *trec*.

Overall, the best scores are obtained with the re-embedded SkGr vectors, which yield the best average accuracy, and outperform all the other configurations on two of the five datasets (*trec* and *imdb*). GV6B (dimension 300) has the second best average scores, outperforming all the other configurations on *sports*. Interestingly, embeddings learned from random vectors achieve performance that are often on a par or higher than those given by HLBL, HPCA or CnW initializations. They actually yield the best performance for the two remaining datasets: *comp* and *religion*. On these tasks, RPA does not seem to benefit from the information contained in the pre-trained embeddings, or their coverage is not just good enough.

For both PA and RPA, performance appear to be positively correlated with embedding coverage: embeddings with lower OOV rates generally perform better those with more missing words. The correlation is only partial, since GV840B do not yield gains compared to GV6B and SkGr despite its better word coverage. Also, SkGr largely outperforms HPCA although they have similar OOV

Emb/Task		comp		religion		sports		trec		imdb		Average	
Method	Size	PA	RPA	PA	RPA	PA	RPA	PA	RPA	PA	RPA	PA	RPA
rand	50	56.76	84.04	59.55	90.24	64.20	90.08	53.20	83.60	60.97	85.74	58.94	86.74
	100	63.06	84.17	67.50	91.35	65.58	90.95	61.20	83.00	64.12	85.90	64.29	87.07
	200	64.86	85.07	71.27	91.07	68.59	90.95	59.60	83.40	67.84	85.75	66.43	87.25
cnw	50	50.32	84.81	55.51	90.93	50.63	91.08	61.20	84.20	50.86	84.82	53.70	87.17
	100	50.71	84.81	55.51	91.35	54.15	91.08	66.40	83.60	50.98	85.40	55.55	87.25
	200	50.45	84.04	55.51	90.93	54.77	91.08	65.80	82.60	52.36	85.50	55.78	86.83
HLBL	50	53.15	85.07	59.97	89.82	56.41	90.95	56.60	83.40	62.38	85.66	57.70	86.98
	100	53.80	84.68	61.09	91.21	56.91	90.95	67.60	84.00	67.88	85.64	61.46	87.30
HPCA	50	50.45	85.20	55.51	91.35	52.39	89.95	62.20	83.00	51.02	83.50	54.31	86.60
	100	50.45	85.20	55.51	91.07	49.87	90.08	66.20	83.60	50.51	84.86	54.51	86.96
	200	50.45	85.20	55.51	90.10	49.87	89.82	68.00	82.80	50.38	85.72	54.84	86.73
GV6B	50	50.97	85.07	64.16	91.49	55.28	91.08	57.00	85.00	69.48	85.48	59.38	87.62
	100	50.58	84.94	60.53	89.68	63.82	91.08	58.00	84.60	70.22	85.51	60.63	87.16
	200	51.22	85.20	64.99	91.49	85.05	90.08	58.00	84.80	73.64	85.59	66.58	87.43
	300	56.24	85.33	70.15	89.68	89.07	91.21	72.40	85.20	75.48	85.79	72.67	87.44
GV840B	300	66.02	84.81	77.96	89.68	89.82	91.08	77.80	86.40	77.57	85.73	77.83	87.54
SkGr	300	67.95	82.50	81.59	89.40	95.10	94.60	80.80	88.40	80.93	85.92	81.27	88.16
one-hot		84.56		89.96		90.58		85.80		87.40		87.66	

Table 4: Accuracy results for RPA against standard PA both run for a single iteration.

rates. As for dimensions, embeddings of length 100 and more perform the best, although they involve estimating larger number of parameters, which is a priori difficult given the small sizes of the datasets.

By comparison with previous work on *imdb*, the RPA performance are substantially better than those reported by (Labutov and Lipson, 2013), whose best re-embedding score is 81.15 with CnW. By comparison, our best score with CnW is 86.64, and 88.52 with SkGr, thus closing the gap on (Maas et al., 2011) who report an accuracy of 88.89 using a much more computationally intensive approach specifically tailored to sentiment analysis. Interestingly, (Labutov and Lipson, 2013) show that accuracy can be further improved by concatenating re-embedded and 1-hot representations. This option is also available to us, but we leave it to future work.

Finally, Table 4 report accuracy results for RPA against PA when both algorithms are trained in a genuine online mode, that is with a single pass over the data. As expected, performance drop for the RPA and the PA, but the decreases are comparatively much smaller for the RPA (from 2% to 3%) compared to the PA (from 0.4% to 14%).

6 Conclusion and Future Work

In this paper, we have proposed a new scalable algorithm for learning word representations that

are specifically tailored to a classification objective. This algorithm generalizes the well-known Passive-Aggressive algorithm, and we showed how to extend the regret bounds results of the PA to the RPA when either the weight vector or the embedding matrix is fixed. In addition, we have also provided synthetic and NLP experiments, demonstrating that the good classification performance of RPA.

In future work, we first would like to achieve a more complete analysis of the RPA algorithm when both w and Φ both get updated. Also, we intend to investigate potential exact methods for solving biconvex minimization (Floudas and Viswewaran, 1990), as well as to develop a stochastic version of RPA, thus foregoing running the inner alternate search to convergence. More empirical perspectives include extending the RPA to linguistic structured prediction tasks, better handling of unknown words, and a deeper intrinsic and statistical evaluation of the embeddings learned by the RPA.

Acknowledgments

We thank the three anonymous reviewers for their comments. Pascal Denis was supported by ANR Grant GRASP No. ANR-16-CE33-0011, as well as by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020.

References

- Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL (2)*, pages 809–815.
- Mathieu Blondel, Yotaro Kubo, and Ueda Naonori. 2014. Online passive-aggressive algorithms for non-negative matrix factorization and completion. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 96–104.
- Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.
- Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.
- C. A. Floudas and V. Viswewaran. 1990. A global optimization algorithm (gop) for certain classes of nonconvex npls. i, theory. *Computers & chemical engineering*, 14(12):1397–1417.
- Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. 2007. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407.
- Y. Grandvalet and S. Canu. 2003. Adaptive scaling for feature selection in svms. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.
- Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.
- Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*.
- Rémi Lebret and Ronan Collobert. 2014. Word emdeddings through hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.
- Rémi Lebret, Joël Legrand, and Ronan Collobert. 2013. Is deep learning really necessary for word embeddings? In *NIPS Workshop on Deep Learning*.
- Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.
- Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.
- Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.
- Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proc. of EMNLP*.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.
- Mengqiu Wang and Christopher D Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *IJCNLP*, pages 1285–1291.

Nonsymbolic Text Representation

Hinrich Schütze

CIS, LMU Munich, Germany

inquiries@cislmu.org

Abstract

We introduce the first generic text representation model that is completely nonsymbolic, i.e., it does not require the availability of a segmentation or tokenization method that attempts to identify words or other symbolic units in text. This applies to training the representations as well as to using them in an application. We demonstrate better performance than prior work on entity typing and text denoising.

1 Introduction

Character-level models can be grouped into three classes. (i) **End-to-end models** learn a separate model on the raw character (or byte) input for each task; these models estimate task-specific parameters, but no representation of text that would be usable across tasks is computed. Throughout this paper, we refer to $r(x)$ as the “representation” of x only if $r(x)$ is a generic rendering of x that can be used in a general way, e.g., across tasks and domains. The activation pattern of a hidden layer for a given input sentence in a multilayer perceptron (MLP) is not a representation according to this definition if it is not used outside of the MLP. (ii) **Character-level models of words** derive a representation of a word w from the character string of w , but they are symbolic in that they need text segmented into tokens as input. (iii) **Bag-of-character-ngram models, bag-of-ngram models** for short, use character ngrams to encode sequence-of-character information, but sequence-of-ngram information is lost in the representations they produce.

Our premise is that text representations are needed in NLP. A large body of work on word embeddings demonstrates that a generic text representation, trained in an unsupervised fashion on

large corpora, is useful. Thus, we take the view that **group (i) models**, end-to-end learning without any representation learning, is not a good general approach for NLP.

We distinguish **training** and **utilization** of the text representation model. We use “training” to refer to the method by which the model is learned and “utilization” to refer to the application of the model to a piece of text to compute a representation of the text. In many text representation models, utilization is trivial. For example, for word embedding models, utilization amounts to a simple lookup of a word to get its precomputed embedding. However, for the models we consider, utilization is not trivial and we will discuss different approaches.

Both training and utilization can be either **symbolic** or **nonsymbolic**. We define a symbolic approach as one that is based on tokenization, i.e., a segmentation of the text into tokens. Symbol identifiers (i.e., tokens) can have internal structure – a tokenizer may recognize tokens like “to and fro” and “London-based” that contain delimiters – and may be morphologically analyzed downstream.¹

We define a nonsymbolic approach as one that is tokenization-free, i.e., no assumption is made that there are segmentation boundaries and that each segment (e.g., a word) should be represented (e.g., by a word embedding) in a way that is independent of the representations (e.g., word embeddings) of neighboring segments. Methods for training text representation models that require tokenized text include word embedding models like word2vec (Mikolov et al., 2013) and most **group**

¹The position-embedding representation of a text introduced below is a sequence of position embeddings. An embedding that represents a single character must be viewed as symbolic since a character is a symbol – just like a representation of text as a sequence of word embeddings is symbolic since each word corresponds to a symbol. But position embeddings do not represent single characters. See §4.

(ii) **methods**, i.e., character-level models like fast-Text skipgram (Bojanowski et al., 2016).

Bag-of-gram models, group (iii) models, are text representation utilization models that typically compute the representation of a text as the sum of the embeddings of all character ngrams occurring in it, e.g., WordSpace (Schütze, 1992) and CHARAGRAM (Wieting et al., 2016). WordSpace and CHARAGRAM are examples of mixed training-utilization models: training is performed on tokenized text (words and phrases), utilization is nonsymbolic.

We make two contributions in this paper. (i) We propose the first generic method for training text representation models without the need for tokenization and address the challenging sparseness issues that make this difficult. (ii) We propose the first nonsymbolic utilization method that fully represents sequence information – in contrast to utilization methods like bag-of-ngrams that discard sequence information that is not directly encoded in the character ngrams themselves.

2 Motivation

Chung et al. (2016) give two motivations for their work on character-level models. First, **tokenization** (or, equivalently, segmentation) **algorithms make many mistakes** and are brittle: “we do not have a perfect word segmentation algorithm for any one language”. Tokenization errors then propagate throughout the NLP pipeline.

Second, there is currently **no general solution for morphology** in statistical NLP. For many languages, high-coverage and high-quality morphological resources are not available. Even for well resourced languages, problems like ambiguity make morphological processing difficult; e.g., “rung” is either the singular of a noun meaning “part of a ladder” or the past participle of “to ring”. In many languages, e.g., in German, syncretism, a particular type of systematic morphological ambiguity, is pervasive. Thus, there is no simple morphological processing method that would produce a representation in which all inflected forms of “to ring” are marked as having a common lemma; and no such method in which an unseen form like “aromatizing” is reliably analyzed as a form of “aromatize” whereas an unseen form like “anti-trafficking” is reliably analyzed as the compound “anti+trafficking”.

Of course, it is an open question whether non-

symbolic methods can perform better than morphological analysis, but the foregoing discussion motivates us to investigate them.

Chung et al. (2016) focus on problems with the tokens produced by segmentation algorithms. Equally important is the problem that **tokenization fails to capture structure across multiple tokens**. The job of dealing with cross-token structure is often given to downstream components of the pipeline, e.g., components that recognize multiwords and named entities in English or in fact any word in a language like Chinese that uses no overt delimiters. However, there is no linguistic or computational reason in principle why we should treat the recognition of a unit like “electromechanical” (containing no space) as fundamentally different from the recognition of a unit like “electrical engineering” (containing a space). Character-level models offer the potential of uniform treatment of such linguistic units.

3 Text representation model: Training

3.1 Methodology

Many text representation learning algorithms can be understood as estimating the parameters of the model from a unit-context matrix C where each row corresponds to a unit u_i , each column to a context c_j and each cell C_{ij} measures the degree of association between u_i and c_j . For example, the skipgram model is closely related to an SVD factorization of a pointwise mutual information matrix (Levy and Goldberg, 2014). Many text representation learning algorithms are formalized as matrix factorization (e.g., (Deerwester et al., 1990; Hofmann, 1999; Stratos et al., 2015)), but there may be no big difference between implicit (e.g., (Pennington et al., 2014)) and explicit factorization methods; see also (Mohamed, 2011; Rastogi et al., 2015).

Our goal in this paper is not to develop new matrix factorization methods. Instead, we will focus on defining the unit-context matrix in such a way that no symbolic assumption has to be made. This unit-context matrix can then be processed by any existing or still to be invented algorithm.

Definition of units and contexts. How to define units and contexts without relying on segmentation boundaries? In initial experiments, we simply generated all character ngrams of length up to k_{\max} (where k_{\max} is a parameter), including character ngrams that cross token boundaries; i.e., no

segmentation is needed. We then used a skipgram-type objective for learning embeddings that attempts to predict, from ngram g_1 , an ngram g_2 in g_1 's context. Results were poor because many training instances consist of pairs (g_1, g_2) in which g_1 and g_2 overlap, e.g., one is a subsequence of the other. So the objective encourages trivial predictions of ngrams that have high string similarity with the input and nothing interesting is learned.

In this paper, we propose an alternative way of defining units and contexts that supports well-performing nonsymbolic text representation learning: **multiple random segmentation**. A pointer moves through the training corpus. The current position i of the pointer defines the left boundary of the next segment. The length l of the next move is uniformly sampled from $[k_{\min}, k_{\max}]$ where k_{\min} and k_{\max} are the minimum and maximum segment lengths. The right boundary of the segment is then $i+l$. Thus, the segment just generated is $c_{i,i+l}$, the subsequence of the corpus between (and including) positions i and $i+l$. The pointer is positioned at $i+l+1$, the next segment is sampled and so on. An example of a random segmentation from our experiments is “@he@had@b egu n@to@show @his@cap acity@f” where space was replaced with “@” and the next segment starts with “or@”.

The corpus is segmented this way m times (where m is a parameter) and the m random segmentations are concatenated. The unit-context matrix is derived from this concatenated corpus.

Multiple random segmentation has two advantages. First, there is no redundancy since, in any given random segmentation, two ngrams do not overlap and are not subsequences of each other. Second, a single random segmentation would only cover a small part of the space of possible ngrams. For example, a random segmentation of “a rose is a rose is a rose” might be “[a ros][e is a ros][e is][a rose]”. This segmentation does not contain the segment “rose” and this part of the corpus can then not be exploited to learn a good embedding for the fourgram “rose”. However, with multiple random segmentation, it is likely that this part of the corpus does give rise to the segment “rose” in one of the segmentations and can contribute information to learning a good embedding for “rose”.

We took the idea of random segmentation from work on biological sequences (Asgari and Mofrad, 2015; Asgari and Mofrad, 2016). Such sequences have no delimiters, so they are a good model if

one believes that delimiter-based segmentation is problematic for text.

3.2 Ngram equivalence classes/Permutation

Form-meaning homomorphism premise. Nonsymbolic representation learning does not preprocess the training corpus by means of tokenization and considers many ngrams that would be ignored in tokenized approaches because they span token boundaries. As a result, the number of ngrams that occur in a corpus is an order of magnitude larger for tokenization-free approaches than for tokenization-based approaches. See supplementary for details.

We will see below that this sparseness impacts performance of nonsymbolic text representation negatively. We address sparseness by defining ngram equivalence classes. All ngrams in an equivalence class receive the same embedding.

The relationship between form and meaning is mostly arbitrary, but there are substructures of the ngram space and the embedding space that are systematically related by homomorphism. In this paper, **we will assume the following homomorphism:**

$$g_1 \sim_{\tau} g_2 \Leftrightarrow \vec{v}(g_1) \sim_{=} \vec{v}(g_2)$$

where $g_1 \sim_{\tau} g_2$ iff $\tau(g_1) = \tau(g_2)$ for string transduction τ and $\vec{v}(g_1) \sim_{=} \vec{v}(g_2)$ iff $|\vec{v}(g_1) - \vec{v}(g_2)|_2 < \epsilon$.

As a simple example consider a transduction τ that deletes spaces at the beginning of ngrams, e.g., $\tau(@Mercedes) = \tau(Mercedes)$. This is an example of a meaning-preserving τ since for, say, English, τ will not change meaning. We will propose a procedure for learning τ below.

We define $\sim_{=}$ as “closeness” – not as identity – because of estimation noise when embeddings are learned. We assume that there are no true synonyms and therefore the direction $g_1 \sim_{\tau} g_2 \Leftarrow \vec{v}(g_1) \sim_{=} \vec{v}(g_2)$ also holds. For example, “car” and “automobile” are considered synonyms, but we assume that their embeddings are different because only “car” has the literary sense “chariot”. If they were identical, then the homomorphism would not hold since “car” and “automobile” cannot be converted into each other by any plausible meaning-preserving τ .

Learning procedure. To learn τ , we define three templates that transform one ngram into another: (i) replace character a_1 with character a_2 ,

(ii) delete character a_1 if its immediate predecessor is character a_2 , (iii) delete character a_1 if its immediate successor is character a_2 . The learning procedure takes a set of ngrams and their embeddings as input. It then exhaustively searches for all pairs of ngrams, for all pairs of characters a_1/a_2 , for each of the three templates. When two matching embeddings exist, we compute their cosine. For example, for the operation “delete space before M”, an ngram pair from our embeddings that matches is “@Mercedes” / “Mercedes” and we compute its cosine. As the characteristic statistic of an operation we take the average of all cosines; e.g., for “delete space before M” the average cosine is .7435. We then rank operations according to average cosine and take the first N_o as the definition of τ where N_o is a parameter. For characters that are replaced by each other (e.g., 1, 2, 3 in Table 1), we compute the equivalence class and then replace the learned operations with ones that replace a character by the canonical member of its equivalence class (e.g., $2 \rightarrow 1, 3 \rightarrow 1$).

Permutation premise. Tokenization algorithms can be thought of as assigning a particular function or semantics to each character and making tokenization decisions accordingly; e.g., they may disallow that a semicolon, the character “;”, occurs inside a token. If we want to learn representations from the data without imposing such hard constraints, then characters should not have any particular function or semantics. A consequence of this desideratum is that if any two characters are exchanged for each other, this should not affect the representations that are learned. For example, if we interchange space and “A” throughout a corpus, then this should have no effect on learning: what was the representation of “NATO” before, should now be the representation of “N TO”. We can also think of this type of permutation as a sanity check: it ensures we do not inadvertently make use of text preprocessing heuristics that are pervasive in NLP.²

Let A be the alphabet of a language, i.e., its set of characters, π a permutation on A , C a corpus and $\pi(C)$ the corpus permuted by π . For example, if $\pi(a) = e$, then all “a” in C are replaced with “e” in $\pi(C)$. **The learning procedure should learn**

²An example of such an inadvertent use of text preprocessing heuristics is that fastText seems to default to lowercase ngrams if embeddings of uppercase ngrams are not available: when fastText is trained on lowercased text and then applied to uppercased text, it still produces embeddings.

identical equivalence classes on C and $\pi(C)$. So, if $g_1 \sim_\tau g_2$ after running the learning procedure on C , then $\pi(g_1) \sim_\tau \pi(g_2)$ after running the learning procedure on $\pi(C)$.

This premise is motivated by our desire to come up with a general method that does not rely on specific properties of a language or genre; e.g., the premise rules out exploiting the fact through feature engineering that in many languages and genres, “c” and “C” are related. Such a relationship has to be learned from the data.

3.3 Experiments

We run experiments on C , a 3 gigabyte English Wikipedia corpus, and train word2vec skipgram (W2V, (Mikolov et al., 2013)) and fastText skipgram (FTX, (Bojanowski et al., 2016)) models on C and its derivatives. We randomly generate a permutation π on the alphabet and learn a transduction τ (details below). In Table 2 (left), the columns “method”, π and τ indicate the method used (W2V or FTX) and whether experiments in a row were run on C , $\pi(C)$ or $\tau(\pi(C))$. The values of “whitespace” are: (i) ORIGINAL (whitespace as in the original), (ii) SUBSTITUTE (what π outputs as whitespace is used as whitespace, i.e., $\pi^{-1}(\text{“ ”})$ becomes the new whitespace) and (iii) RANDOM (random segmentation with parameters $m = 50, k_{\min} = 3, k_{\max} = 9$). Before random segmentation, whitespace is replaced with “@” – this character occurs rarely in C , so that the effect of conflating two characters (original “@” and whitespace) can be neglected. The random segmenter then indicates boundaries by whitespace – unambiguously since it is applied to text that contains no whitespace.

We learn τ on the embeddings learned by W2V on the random segmentation version of $\pi(C)$ (C-RANDOM in the table) as described in §3.2 for $N_o = 200$. Since the number of equivalence classes is much smaller than the number of ngrams, τ reduces the number of distinct character ngrams from 758M in the random segmentation version of $\pi(C)$ (C/D-RANDOM) to 96M in the random segmentation version of $\tau(\pi(C))$ (E/F-RANDOM).

Table 1 shows a selection of the N_o operations. Throughout the paper, if we give examples from $\pi(C)$ or $\tau(\pi(C))$ as we do here, we convert characters back to the original for better readability. The two uppercase/lowercase conversions shown

substitution	$2 \rightarrow 1$	predeletion	$/r \rightarrow r$	postdeletion	$\ddagger@ \rightarrow \ddagger$
	$3 \rightarrow 1$		$@\ddagger \rightarrow \ddagger$		$e@ \rightarrow e$
	$:$		$@\ddagger \rightarrow \ddagger$		$l@ \rightarrow l$
	$;$		$@H \rightarrow H$		$m@ \rightarrow m$
	$E \rightarrow e$		$@I \rightarrow I$		$ml \rightarrow m$
	$C \rightarrow c$				

Table 1: String operations that on average do not change meaning. “@” stands for space. \ddagger is the left or right boundary of the ngram.

in the table ($E \rightarrow e$, $C \rightarrow c$) were the only ones that were learned (we had hoped for more). The post-deletion rule $ml \rightarrow m$ usefully rewrites “html” as “htm”, but is likely to do more harm than good. We inspected all 200 rules and, with a few exceptions like $ml \rightarrow m$, they looked good to us.

Evaluation. We evaluate the three models on an entity typing task, similar to (Yaghoobzadeh and Schütze, 2015), but based on an entity dataset released by Xie et al. (2016) in which each entity has been assigned one or more types from a set of 50 types. For example, the entity “Harrison Ford” has the types “actor”, “celebrity” and “award winner” among others. We extract mentions from FACC (<http://lemurproject.org/clueweb12/FACC1>) if an entity has a mention there or we use the Freebase name as the mention otherwise. This gives us a data set of 54,334, 6085 and 6747 mentions in train, dev and test, respectively. Each mention is annotated with the types that its entity has been assigned by Xie et al. (2016). The evaluation has a strong cross-domain aspect because of differences between FACC and Wikipedia, the training corpus for our representations. For example, of the 525 mentions in dev that have a length of at least 5 and do not contain lowercase characters, more than half have 0 or 1 occurrences in the Wikipedia corpus, including many like “JOHNNY CARSON” that are frequent in other case variants.

Since our goal in this experiment is to evaluate tokenization-free learning, not tokenization-free utilization, we use a simple utilization baseline, the bag-of-ngram model (see §1). A mention is represented as the sum of all character ngrams that embeddings were learned for. Linear SVMs (Chang and Lin, 2011) are then trained, one for each of the 50 types, on train and applied to dev and test. Our evaluation measure is micro F_1 on all typing decisions; e.g., one typing decision is:

“Harrison Ford” is a mention of type “actor”. We tune thresholds on dev to optimize F_1 and then use these thresholds on test.

3.4 Results

Results are presented in Table 2 (left). Overall performance of FTX is higher than W2V in all cases. For ORIGINAL, FTX’s recall is a lot higher than W2V’s whereas precision decreases slightly. This indicates that FTX is stronger in both learning and application: in learning it can generalize better from sparse training data and in application it can produce representations for OOVs and better representations for rare words. For English, prefixes, suffixes and stems are of particular importance, but there often is not a neat correspondence between these traditional linguistic concepts and internal FTX representations; e.g., Bojanowski et al. (2016) show that “asphal”, “sphalt” and “phalt” are informative character ngrams of “asphaltic”.

Running W2V on random segmentations can be viewed as an alternative to the learning mechanism of FTX, which is based on character ngram cooccurrence; so it is not surprising that for RANDOM, FTX has only a small advantage over W2V.

For C/D-SUBSTITUTE, we see a dramatic loss in performance if tokenization heuristics are not used. This is not surprising, but shows how powerful tokenization can be.

C/D-ORIGINAL is like C/D-SUBSTITUTE except that we artificially restored the space – so the permutation π is applied to all characters except for space. By comparing C/D-ORIGINAL and C/D-SUBSTITUTE, we see that the space is the most important text preprocessing feature employed by W2V and FTX. If space is restored, there is only a small loss of performance compared to A/B-ORIGINAL. So text preprocessing heuristics other than whitespace tokenization in a narrow definition of the term (e.g., downcasing) do not seem to play a big role, at least not for our entity typing task.

For tokenization-free embedding learning on random segmentation, there is almost no difference between original data (A/B-RANDOM) and permuted data (C/D-RANDOM). This confirms that our proposed learning method is insensitive to permutations and makes no use of text preprocessing heuristics.

We achieve an additional improvement by applying the transduction τ . In fact, FTX perfor-

		whitespace	ORIGINAL			SUBSTITUTE			RANDOM			query	neighbor	r	
		measure	P	R	F_1	P	R	F_1	P	R	F_1				
method	π	τ													
A W2V	–	–	.538	.566	.552				.525	.596	.558	1	Abdulaziz	Abdul Azi	2
B FTX	–	–	.530	.628	.575				.528	.608	.565	2	codenamed	code name	1
C W2V	+	–	.535	.560	.547	.191	.296	.233	.514	.605	.556	3	Quarterfi	uarter-Fi	1
D FTX	+	–	.530	.623	.573	.335	.510	.405	.531	.608	.567	4	worldreco	orld-reco	1
E W2V	+	+							.503	.603	.548	5	antibodie	stem cell	1
F FTX	+	+							.551	.618	.582	6	eflectors	ear wheel	1
												7	ommandeer	rash land	1
												8	reenplays	ripts for	1
												9	roughfare	ugh downt	1
												10	ilitating	e-to-face	1

Table 2: Left: Evaluation results for named entity typing. Right: Neighbors of character ngrams. Rank $r = 1/r = 2$: nearest / second-nearest neighbor.

mance for F-RANDOM (F_1 of .582) is better than tokenization-based W2V and FTX performance. Thus, our proposed method seems to be an effective tokenization-free alternative to tokenization-based embedding learning.

3.5 Analysis of ngram embeddings

Table 2 (right) shows nearest neighbors of ten character ngrams, for the A-RANDOM space. Queries were chosen to contain only alphanumeric characters. To highlight the difference to symbol-based representation models, we restricted the search to 9-grams that contained a delimiter at positions 3, 4, 5, 6 or 7.

Lines 1–4 show that “delimiter variation”, i.e., cases where a word has two forms, one with a delimiter, one without a delimiter, is handled well: “Abdulaziz” / “Abdul Azi”, “codenamed” / “code name”, “Quarterfinal” / “Quarter-Final”, “world-record” / “world-record”.

Lines 5–9 are cases of ambiguous or polysemous words that are disambiguated through “character context”. “stem”, “cell”, “rear”, “wheel”, “crash”, “land”, “scripts”, “through”, “downtown” all have several meanings. In contrast, the meanings of “stem cell”, “rear wheel”, “crash land”, “(write) scripts for” and “through downtown” are less ambiguous. A multiword recognizer may find the phrases “stem cell” and “crash land” automatically. But the examples of “scripts for” and “through downtown” show that what is accomplished here is not multiword detection, but a more general use of character context for disambiguation.

Line 10 shows that a 9-gram of “face-to-face” is the closest neighbor to a 9-gram of “facilitating”. This demonstrates that form and meaning sometimes interact in surprising ways. Facilitating a meeting is most commonly done face-to-face. It is not inconceivable that form – the shared trigram “fac” or the shared fourgram “faci” in “facilitate”

/ “facing” – is influencing meaning here in a way that also occurs historically in cases like “ear” ‘organ of hearing’ / “ear” ‘head of cereal plant’, originally unrelated words that many English speakers today intuit as one word.

4 Utilization: Tokenization-free representation of text

4.1 Methodology

The main text representation model that is based on ngram embeddings similar to ours is the **bag-of-ngram model**. A sequence of characters is represented by a single vector that is computed as the sum of the embeddings of all ngrams that occur in the sequence. In fact, this is what we did in the entity typing experiment. In most work on bag-of-ngram models, the sequences considered are words or phrases (see (Schuetze, 2016) for citations). In a few cases, the model is applied to longer sequences, including sentences and documents; e.g., (Schütze, 1992), (Wieting et al., 2016).

The basic assumption of the bag-of-ngram model is that sequence information is encoded in the character ngrams and therefore a “bag-of” approach (which usually throws away all sequence information) is sufficient. The assumption is not implausible: for most bags of character sequences, there is only a single way of stitching them together to one coherent sequence, so in that case information is not necessarily lost (although this is likely when embeddings are added). But the assumption has not been tested experimentally.

Here, we propose **position embeddings**, character-ngram-based embeddings that more fully preserve sequence information.³ The simple idea is to represent each position as the sum of all ngrams that contain that position. When we set

³Position embeddings were independently proposed by Kalchbrenner et al. (2016), see Section 3.6 of their paper.

POS		$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$
2	e	wealthies	accolades	bestselle	bestselli	Billboard
3	s	estseller	wealthies	bestselli	accolades	bestselle
15	o	fortnight	afternoon	overnight	allowance	Saturdays
16	n	fortnight	afternoon	Saturdays	Wednesday	magazines
23	o	superhero	ntagraphi	adventure	Astonishi	bestselli
24	m	superhero	ntagraphi	anthology	Daredevil	Astonishi
29	o	anthology	paperback	superhero	Lovecraft	tagraphic
30	o	anthology	paperback	tagraphic	Lovecraft	agraphics
34	u	antagraph	agraphics	paperback	hardcover	ersweekly
35	b	ublishing	ublishers	ublicatio	antagraph	aperbacks

Table 3: Nearest ngram embeddings (rank $r \in [1, 5]$) of the position embeddings for “POS”, the positions 2/3 (best), 15/16 (monthly), 23/24 (comic), 29/30 (book) and 34/35 (publications) in the Wikipedia excerpt “best-selling monthly comic book publications sold in North America”

$k_{\min} = 3$, $k_{\max} = 9$, this means that the position is the sum of $(\sum_{3 \leq k \leq 9} k)$ ngram embeddings (if all of these ngrams have embeddings, which generally will be true for some, but not for most positions). A sequence of n characters is then represented as a sequence of n such position embeddings.

4.2 Experiments

We again use the embeddings corresponding to A-RANDOM in Table 2. We randomly selected 2,000,000 contexts of size 40 characters from Wikipedia. We then created a noise context for each of the 2,000,000 contexts by replacing one character at position i ($15 \leq i \leq 25$, uniformly sampled) with space (probability $p = .5$) or a random character otherwise. Finally, we selected 1000 noise contexts randomly and computed their nearest neighbors among the 4,000,000 contexts (excluding the noise query). We did this in two different conditions: for a bag-of-ngram representation of the context (sum of all character ngrams) and for the concatenation of 11 position embeddings, those between 15 and 25. Our evaluation measure is mean reciprocal rank of the clean context corresponding to the noise context. This simulates a text denoising experiment: if the clean context has rank 1, then the noisy context can be corrected.

Table 4 shows that sequence-preserving position embeddings perform better than bag-of-

	bag-of-ngram	position embeddings
MRR	.64	.76

Table 4: Mean reciprocal rank of text denoising experiment for bag-of-ngram text representation and position embedding text representation

	exchange@f (in exchange for)	ic@exchang (many contexts)	ing@exchan (many contexts)
exchange@f	1.000	0.008	-0.056
ic@exchang	0.008	1.000	0.108
ing@exchan	-0.056	0.108	1.000

	xchange@ra (exchange rates)	ival@rates (survival rates)	rime@rates (crime rates)
xchange@ra	1.000	0.036	0.050
ival@rates	0.036	1.000	0.331
rime@rates	0.050	0.331	1.000

Table 6: Cosine similarity of ngrams that cross word boundaries and disambiguate polysemous words. The tables show three disambiguating ngrams for “exchange” and “rates” that have different meanings as indicated by low cosine similarity. In phrases like “floating exchange rates” and “historic exchange rates”, disambiguating ngrams overlap. Parts of the word “exchange” are disambiguated by preceding context (ic@exchang, ing@exchan) and parts of “exchange” provide context for disambiguating “rates” (xchange@ra).

ngram representations.

Table 5 shows an example of a context in which position embeddings did better than bag-of-ngrams, demonstrating that sequence information is lost by bag-of-ngram representations, in this case the exact position of “Seahawks”.

Table 3 gives further intuition about the type of information position embeddings contain, showing the ngram embeddings closest to selected position embeddings; e.g., “estseller” (the first 9-gram on the line numbered 3 in the table) is closest to the embedding of position 3 (corresponding to the first “s” of “best-selling”). The kNN search space is restricted to alphanumeric ngrams.

5 Discussion

Single vs. multiple segmentation. The motivation for multiple segmentation is *exhaustive cov-*

	rep. space	similarity	r	left context	center	right context
1	correct			s and Seattle S	eahawks th	at led to publi
2	noise (query)			s and Seattle S	eahawks t	at led to publi
3	position-emb	.761	1	s and Seattle S	eahawks th	at led to publi
4	bag-of-ngram	.904	1	arted 15 games	fsr the Se	ahawks, leading
5	bag-of-ngram	.864	6	s and Seattle S	eahawks th	at led to publi

Table 5: Illustration of the result in Table 4. “rep. space” = “representation space”. We want to correct the error in the corrupted “noise” context (line 2) and produce “correct” (line 1). The nearest neighbor to line 2 in position-embedding space is the correct context (line 3, $r = 1$). The nearest neighbor to line 2 in bag-of-ngram space is incorrect (line 4, $r = 1$) because the precise position of “Seahawks” in the query is not encoded. The correct context in bag-of-ngram space is instead at rank $r = 6$ (line 5). “similarity” is average cosine (over eleven position embeddings) for position embeddings.

erage of the space of possible segmentations. An alternative approach would be to attempt to find a single optimal segmentation.

Our intuition is that in many cases *overlapping segments contain complementary information*. Table 6 gives an example. Historic exchange rates are different from floating exchange rates and this is captured by the low similarity of the ngrams `ic@exchang` and `ing@exchan`. Also, the meaning of “historic” and “floating” is non-compositional: these two words take on a specialized meaning in the context of exchange rates. The same is true for “rates”: its meaning is not its general meaning in the compound “exchange rates”. Thus, we need a representation that contains overlapping segments, so that “historic” / “floating” and “exchange” can disambiguate each other in the first part of the compound and “exchange” and “rates” can disambiguate each other in the second part of the compound. A single segmentation cannot capture these overlapping ngrams.

What text-type are tokenization-free approaches most promising for? The reviewers thought that language and text-type were badly chosen for this paper. Indeed, a morphologically complex language like Turkish and a noisy text-type like Twitter would seem to be better choices for a paper on robust text representation.

However, robust word representation methods like FTX are effective for *within-token* generalization, in particular, effective for both complex morphology and OOVs. If linguistic variability and noise only occur on the token level, then a tokenization-free approach has fewer advantages.

On the other hand, the foregoing discussion of *cross-token* regularities and disambiguation applies to well-edited English text as much as it does to other languages and other text-types as the example of “exchange” shows (which is dis-

ambiguated by prior context and provides disambiguating context to following words) and as is also exemplified by lines 5–9 in Table 2 (right).

Still, this paper does not directly evaluate the different contributions that within-token character ngram embeddings vs. cross-token character ngram embeddings make, so this is an open question. One difficulty is that few corpora are available that allow the separate evaluation of white-space tokenization errors; e.g., OCR corpora generally do not distinguish a separate class of white-space tokenization errors.

Position embeddings vs. phrase/sentence embeddings. Position embeddings may seem to stand in opposition to phrase/sentence embeddings. For many tasks, we need a fixed length representation of a longer sequence; e.g., sentiment analysis models compute a fixed-length representation to classify a sentence as positive / negative.

To see that position embeddings are compatible with fixed-length embeddings, observe first that, in principle, there is *no difference between word embeddings and position embeddings* in this respect. Take a sequence that consists of, say, 6 words and 29 characters. The initial representation of the sentence has length 6 for word embeddings and length 29 for position embeddings. In both cases, we need a model that reduces the variable length sequence into a fixed length vector at some intermediate stage and then classifies this vector as positive or negative. For example, both word and position embeddings can be used as the input to an LSTM whose final hidden unit activations are a fixed length vector of this type.

So assessing position embeddings is not a question of variable-length vs. fixed-length representations. Word embeddings give rise to variable-length representations too. The question is solely whether the position-embedding representation is

a more effective representation.

A more specific form of this argument concerns architectures that compute fixed-length representations of subsequences on intermediate levels, e.g., CNNs. The difference between position-embedding-based CNNs and word-embedding-based CNNs is that the former have access to a *vastly increased range of subsequences*, including substrings of words (making it easier to learn that “exchange” and “exchanges” are related) and cross-token character strings (making it easier to learn that “exchange rate” is noncompositional). Here, the questions are: (i) how useful are subsequences made available by position embeddings and (ii) is the increased level of noise and decreased efficiency caused by many useless subsequences worth the information gained by adding useful subsequences.

Independence of training and utilization.

We note that our proposed training and utilization methods are completely independent. Position embeddings can be computed from any set of character-ngram-embeddings (including FTX) and our character ngram learning algorithm could be used for applications other than position embeddings, e.g., for computing word embeddings.

Context-free vs. context-sensitive embeddings. Word embeddings are context-free: a given word w like “king” is represented by the same embedding independent of the context in which w occurs. Position embeddings are context-free as well: if the maximum size of a character ngram is k_{\max} , then the position embedding of the center of a string s of length $2k_{\max} - 1$ is the same independent of the context in which s occurs.

It is conceivable that text representations could be context-sensitive. For example, the hidden states of a character language model have been used as a kind of nonsymbolic text representation (Chrupala, 2013; Evang et al., 2013; Chrupala, 2014) and these states are context-sensitive. However, such models will in general be a second level of representation; e.g., the hidden states of a character language model generally use character embeddings as the first level of representation. Conversely, position embeddings can also be the basis for a context-sensitive second-level text representation. We have to start somewhere when we represent text. Position embeddings are motivated by the desire to provide a representation that can be computed easily and quickly (i.e., without taking

context into account), but that on the other hand is much richer than the symbolic alphabet.

Processing text vs. speech vs. images. Gillick et al. (2016) write: “It is worth noting that noise is often added . . . to images . . . and speech where the added noise does not fundamentally alter the input, but rather blurs it. [bytes allow us to achieve] something like blurring with text.” It is not clear to what extent blurring on the byte level is useful; e.g., if we blur the bytes of the word “university” individually, then it is unlikely that the noise generated is helpful in, say, providing good training examples in parts of the space that would otherwise be unexplored. In contrast, the text representation we have introduced in this paper can be blurred in a way that is analogous to images and speech. Each embedding of a position is a vector that can be smoothly changed in every direction. We have showed that the similarity in this space gives rise to natural variation.

Prospects for completely tokenization-free processing.

We have focused on whitespace tokenization and proposed a whitespace-tokenization-free method that computes embeddings of higher quality than tokenization-based methods. However, there are many properties of edited text beyond whitespace tokenization that a complex rule-based tokenizer exploits. In a small explorative experiment, we replaced all non-alphanumeric characters with whitespace and repeated experiment A-ORIGINAL for this setting. This results in an F_1 of .593, better by .01 than the best tokenization-free method. This illustrates that there is still a lot of work to be done before we can obviate the need for tokenization.

6 Conclusion

We introduced the first generic text representation model that is completely nonsymbolic, i.e., it does not require the availability of a segmentation or tokenization method that identifies words or other symbolic units in text. This is true for the training of the model as well as for applying it when computing the representation of a new text. In contrast to prior work that has assumed that the sequence-of-character information captured by character ngrams is sufficient, position embeddings also capture sequence-of-ngram information. We showed that our model performs better than prior work on entity typing and text denoising.

References

- Ehsaneddin Asgari and Mohammad R. K. Mofrad. 2015. Protvec: A continuous distributed representation of biological sequences. *CoRR*, abs/1503.05140.
- Ehsaneddin Asgari and Mohammad R. K. Mofrad. 2016. Comparing fifty natural languages and twelve genetic languages using word embedding language divergence (WELD) as a quantitative measure of language distance. *CoRR*, abs/1604.08561.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.
- Chih-Chung Chang and Chih-Jen Lin. 2011. LIB-SVM: A library for support vector machines. *ACM TIST*, 2(3):27:1–27:27.
- Grzegorz Chrupala. 2013. Text segmentation with character-level text embeddings. *CoRR*, abs/1309.4628.
- Grzegorz Chrupala. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 680–686. The Association for Computer Linguistics.
- Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.
- Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.
- Kilian Evang, Valerio Basile, Grzegorz Chrupala, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1422–1426. ACL.
- Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1296–1306. The Association for Computational Linguistics.
- Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 50–57. ACM.
- Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.
- Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.
- Shakir Mohamed. 2011. *Generalised Bayesian matrix factorisation models*. Ph.D. thesis, University of Cambridge, UK.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.
- Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: representation learning via generalized CCA. In Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 556–566. The Association for Computational Linguistics.
- Hinrich Schuetze. 2016. Nonsymbolic text representation. *CoRR*, abs/1610.00479.
- Hinrich Schütze. 1992. Word space. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 895–902. Morgan Kaufmann.

- Karl Stratos, Michael Collins, and Daniel J. Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1282–1291. The Association for Computer Linguistics.
- John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1504–1515. The Association for Computational Linguistics.
- Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2659–2665. AAAI Press.
- Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 715–725. The Association for Computational Linguistics.

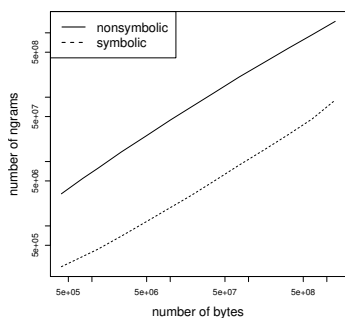


Figure 1: The graph shows how many different character ngrams ($k_{\min} = 3$, $k_{\max} = 10$) occur in the first n bytes of the English Wikipedia for symbolic (tokenization-based) vs. nonsymbolic (tokenization-free) processing. The number of ngrams is an order of magnitude larger in the nonsymbolic approach. We counted all segments, corresponding to $m = \infty$. For the experiments in the paper ($m = 50$), the number of nonsymbolic character ngrams is smaller.

A Supplementary material

A.1 Related work

The related work section appears in the long version of this paper (Schuetze, 2016).

A.2 Acknowledgments

This work was supported by DFG (SCHUE 2246/10-1) and Volkswagenstiftung. We are grateful for their comments to: the anonymous reviewers, Ehsan Asgari, Annemarie Friedrich, Helmut Schmid, Martin Schmitt and Yadollah Yaghoobzadeh.

A.3 Sparseness in tokenization-free approaches

Nonsymbolic representation learning does not preprocess the training corpus by means of tokenization and considers many ngrams that would be ignored in tokenized approaches because they span token boundaries. As a result, the number of ngrams that occur in a corpus is an order of magnitude larger for tokenization-free approaches than for tokenization-based approaches. See Figure 1.

A.4 Experimental settings

W2V hyperparameter settings. size of word vectors: 200, max skip length between words: 5, threshold for occurrence of words: 0, hierarchical softmax: 0, number of negative examples: 5,

threads: 50, training iterations: 1, min-count: 5, starting learning rate: .025, classes: 0

FTX hyperparameter settings. learning rate: .05, lrUpdateRate: 100, size of word vectors: 200, size of context window: 5, number of epochs: 1, minimal number of word occurrences: 5, number of negatives sampled: 5, max length of word ngram: 1, loss function: ns, number of buckets: 2,000,000, min length of char ngram: 3, max length of char ngram: 6, number of threads: 50, sampling threshold: .0001

We ran some experiments with more epochs, but this did not improve the results.

A.5 Other hyperparameters

We did not tune $N_o = 200$, but results are highly sensitive to the value of this parameter. If N_o is too small, then beneficial conflations (collapse punctuation marks, replace all digits with one symbol) are not found. If N_o is too large, then precision suffers – in the extreme case all characters are collapsed into one.

We also did not tune $m = 50$, but we do not consider results to be very sensitive to the value of m if it is reasonably large. Of course, if a larger range of character ngram lengths is chosen, i.e., a larger interval $[k_{\min}, k_{\max}]$, then at some point $m = 50$ will not be sufficient and possible segmentations would not be covered well enough in sampling.

The type of segmentation used in multiple segmentation can also be viewed as a hyperparameter. An alternative to random segmentation would be exhaustive segmentation, but a naive implementation of that strategy would increase the size of the training corpus by several orders of magnitude. Another alternative is to choose one fixed size, e.g., 4 or 5 (similar to (Schütze, 1992)). Many of the nice disambiguation effects we see in Table 2 (right) and in Table 6 would not be possible with short ngrams. On the other hand, a fixed ngram size that is larger, e.g., 10, would make it difficult to get 100% coverage: there would be positions for which no position embedding can be computed.

Fine-Grained Entity Type Classification by Jointly Learning Representations and Label Embeddings

Abhishek, Ashish Anand and Amit Awekar
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Assam, India - 781039

{abhishek.abhishek, anand.ashish, awekar}@iitg.ernet.in

Abstract

Fine-grained entity type classification (FETC) is the task of classifying an entity mention to a broad set of types. Distant supervision paradigm is extensively used to generate training data for this task. However, generated training data assigns same set of labels to every mention of an entity without considering its local context. Existing FETC systems have two major drawbacks: assuming training data to be noise free and use of hand crafted features. Our work overcomes both drawbacks. We propose a neural network model that jointly learns entity mentions and their context representation to eliminate use of hand crafted features. Our model treats training data as noisy and uses non-parametric variant of hinge loss function. Experiments show that the proposed model outperforms previous state-of-the-art methods on two publicly available datasets, namely FIGER(GOLD) and BBN with an average relative improvement of 2.69% in micro-F1 score. Knowledge learnt by our model on one dataset can be transferred to other datasets while using same model or other FETC systems. These approaches of transferring knowledge further improve the performance of respective models.

1 Introduction

Entity type classification is the task for assigning types or labels such as *organization*, *location* to entity mentions in a document. This classification is useful for many natural language processing (NLP) tasks such as relation extraction (Mintz et al., 2009), machine translation (Koehn et al.,

2007), question answering (Lin et al., 2012) and knowledge base construction (Dong et al., 2014).

There has been considerable amount of work on Named Entity Recognition (NER) (Collins and Singer, 1999; Tjong Kim Sang and De Meulder, 2003; Ratnov and Roth, 2009; Manning et al., 2014), which classifies entity mentions into a small set of mutually exclusive types, such as *Person*, *Location*, *Organization* and *Misc*. However, these types are not enough for some NLP applications such as relation extraction, knowledge base construction (KBC) and question answering. In relation extraction and KBC, knowing fine-grained types for entities can significantly increase the performance of the relation extractor (Ling and Weld, 2012; Koch et al., 2014; Mitchell et al., 2015) since this helps in filtering out candidate relation types that do not follow the type constrain. Fine-grained entity types provide additional information while matching questions to its potential answers and significantly improves performance (Dong et al., 2015). For example, Li and Roth (2002) rank questions based on their expected answer types (will the answer be *food*, *vehicle* or *disease*).

Typically, FETC systems use over hundred labels, arranged in a hierarchical structure. An important aspect of FETC is that based on local context, two different mentions of same entity can have different labels. We illustrate this through an example in Figure 1. All three sentences *S1*, *S2*, and *S3* mention same entity *Barack Obama*. However, looking at the context, we can infer that *S1* mentions Obama as a *person/author*, *S2* mentions Obama only as a *person*, and *S3* mentions Obama as a *person/politician*.

Available training data for FETC has noisy labels. Creating manually annotated training data for FETC is time consuming, expensive, and error prone. Note that, a human annotator will

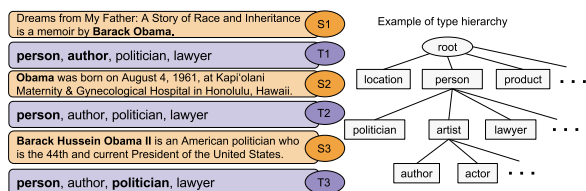


Figure 1: Noise introduced via distant supervision process. S1-S3 indicates sentences where only a subset of labels for entity mention (bold typeface) are relevant given context, highlighted in T1-T3.

have to assign a subset of correct labels from a set of around hundred labels for each entity mention in the corpus. Existing FETC systems use distant supervision paradigm (Craven and Kumlien, 1999) to automatically generate training data. Distant supervision maps each entity in the corpus to knowledge bases such as Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007). This method assigns same set of labels to all mentions of an entity across the corpus. For example, *Barack Obama* is a person, politician, lawyer, and author. If a knowledge base has these four matching labels for Barack Obama, then distant supervision assigns all of them to every mention of Barack Obama. Training data generated with distant supervision will fail to distinguish between mentions of Barack Obama in sentences *S1*, *S2*, and *S3*.

Existing FETC systems have one or both of following drawbacks:

1. Assuming training data to be noise free (Ling and Weld, 2012; Yosef et al., 2012; Yogatama et al., 2015; Shimaoka et al., 2016)
2. Use of hand crafted features (Ling and Weld, 2012; Yosef et al., 2012; Yogatama et al., 2015; Ren et al., 2016)

We have observed that for real world datasets, more than twenty five percent of training data has noisy labels. First drawback propagates this noise in training data to the FETC model. To extract hand crafted features various NLP tools are used. Since errors inevitably exist in such tools, the second drawback propagates errors of these tools to FETC model.

We propose a neural network based model to overcome the two drawbacks of existing FETC systems. First, we separate training data into *clean* and *noisy* partitions using the same method as in AFET system (Ren et al., 2016). For these parti-

tions, we use simple yet effective non-parametric variant of hinge loss function while training. To avoid use of hand crafted features, we learn representations for given entity mention and its context.

Additionally, we investigate effectiveness of using transfer learning (Pratt, 1993) for FETC task both at feature and model level. We show that feature level transfer learning can be used to improve performance of other FETC system such as AFET, by up to 4.5% in micro-F1 score. Similarly, model level transfer learning can be used to improve performance of the same model using different dataset by up to 3.8% in micro-F1 score.

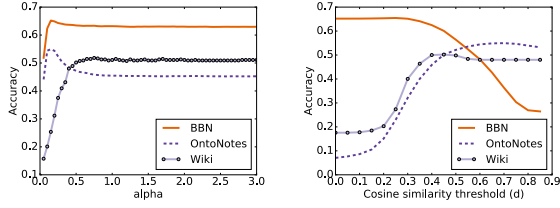
Our contributions can be summarized as follows:

1. We propose a simple neural network model that learns representations for entity mention and its context, and incorporate noisy label information using a variant of non-parametric hinge loss function. Experimental results on two publicly available datasets demonstrate the effectiveness of proposed model, with an average relative improvement of 2.69% in micro-F1 score.
2. We investigate the use of feature level and model level transfer-learning strategies in the domain of the FETC task. The proposed transfer learning strategies further improve the state-of-the-art on BBN dataset by 3.8% in micro-F1 score.

2 Related Work

Ling et al. (2012) proposed the first system for FETC task, which used 112 overlapping labels. They used linear classifier perceptron for multi-label classification. Yosef et al. (2012) used multiple binary SVM classifiers in a hierarchy, to classify an entity mention to a set of 505 types. While the initial work assumed that all labels present in a training dataset for an entity mention are correct, Gillick et al. (2014) introduced context dependent FETC and proposed a set of heuristics for pruning labels that might not be relevant given the entity mention's local context. Yogatama et al. (2015) proposed an embedding based model where user-defined features and labels were embedded into a low dimensional feature space to facilitate information sharing among labels.

Shimaoka et al. (2016) proposed an attentive neural network model that used LSTMs to encode entity mention's context and used an atten-



(a) α models label-label correlation. Higher the α , lower above this threshold are predicted as positive correlation labels.

Figure 2: Effect of change of parameters on AFET’s performance.

tion mechanism to allow the model to focus on relevant expressions in the entity mention’s context. However, the model assumed that all labels obtained via distant supervision are correct. In contrast, our model does not assume that all labels are correct. To learn entity representation, we propose a scheme which is simpler yet more effective.

Most recently, Ren et al. (2016) have proposed AFET, an FETC system. AFET separates the loss function for *clean* and *noisy* entity mentions. AFET uses label-label correlation information obtained by given data in its parametric loss function (model parameter α). During inference, AFET uses a threshold to separate positive types from negative types (similarity threshold parameter d). However, AFET’s loss function is sensitive to change in parameters, which are data dependent. Figure 2 shows the effect of parameter α and d , on AFET performance evaluated on different datasets. In contrast, our model uses a simple yet effective variant of hinge loss function. This function does not need to tune the similarity threshold.

Transfer learning is well applied to many NLP applications, such as cross-domain document classification (Shi et al., 2010), multi-lingual word clustering (Täckström et al., 2012) and sentiment classification (Mou et al., 2016). Initialization of word vectors with pre-trained word vectors in neural network models can be considered as one of the best example of transfer learning in NLP. Wang et al. (2015) provide a broad overview of transfer learning techniques used for language processing.

3 The Proposed Model

3.1 Problem description

Our task is to automatically classify type information of entity mentions present in natural language

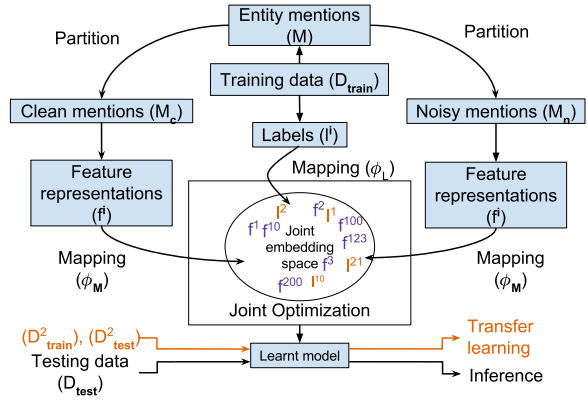


Figure 3: The system overview.

sentences. Figure 3 shows a general overview of our proposed approach.

Input: The input to the model is a training and testing corpus consisting of a set of sentences on which entity mentions have been identified. In training corpus, every entity mention will have corresponding labels according to a given hierarchy. Formally, a training corpus \mathcal{D}_{train} consists of a set of sentences, $\mathcal{S} = \{s^i\}_{i=1}^N$. Each sentence s^i will have one or more entity mentions denoted by $m_{j,k}^i$, where j and k denotes indices of start and end tokens, respectively. Set \mathcal{M} consists of all the entity mentions $m_{j,k}^i$. For every entity mention $m_{j,k}^i$, there will be a corresponding label vector $l_{j,k}^i \in \{0, 1\}^K$, which is a binary vector, where $l_{j,k}^i = 1$ if t^{th} type is true otherwise it will be zero. K denotes the total number of labels in a given hierarchy Ψ . Testing corpus \mathcal{D}_{test} will only contain sentences and entity mentions.

Output: For entity mentions in testing corpus \mathcal{D}_{test} , predict their corresponding labels.

3.2 Training set partition

Similar to AFET, we partition the mention set \mathcal{M} of training corpus \mathcal{D}_{train} into two parts, a set \mathcal{M}_c , consisting only of *clean* entity mentions and a set \mathcal{M}_n , consisting only of *noisy* entity mentions. An entity mention $m_{j,k}^i$ is said to be *clean* if its labels $l_{j,k}^i$ belong to only a single path (not necessary to be leaf) in the hierarchy Ψ , that is its labels are not ambiguous; otherwise, it is *noisy*. For example, as per hierarchy given in figure 1, an entity mention with labels *person*, *artist* and *politician* will be considered as *noisy*, whereas entity mention with labels *person*, *artist* and *actor* will be considered as *clean*.

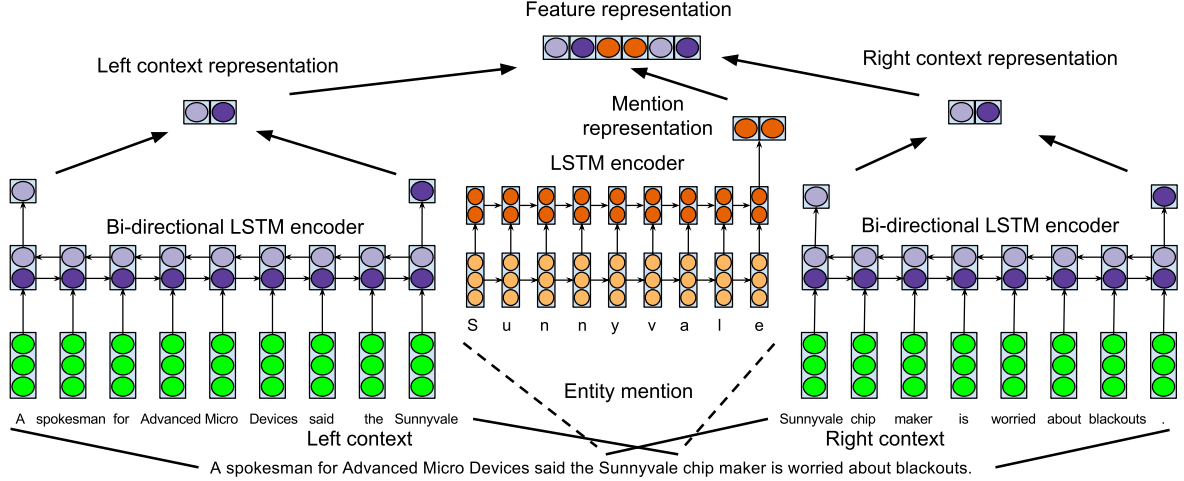


Figure 4: The architecture of the proposed model.

3.3 Feature representations

Mention representation: This representation captures information about entity mention’s morphology and orthography. We decompose an entity mention into character sequence, and use a vanilla LSTM encoder (Hochreiter and Schmidhuber, 1997) to encode character sequences to a fixed dimensional vector. Formally, for entity mention $m_{j,k}^i$, we decompose it into a sequence of character tokens $c_{j,k_1}^i, c_{j,k_2}^i, \dots, c_{j,k_{|m_{j,k}^i|}}^i$, where $|m_{j,k}^i|$ denotes the total number of characters present in the entity mention. For entity mention containing multiple tokens, we join these tokens with a space in between tokens. Every character will have corresponding vector representation in a lookup table for characters. The character sequence is then fed one by one to a LSTM encoder, and the final output is used as a feature representation for entity mention $m_{j,k}^i$. We denote this process by a function $F_m : \mathcal{M} \rightarrow \mathbb{R}^{D_m}$, where D_m is the number of dimensions for mention representation. The whole process is illustrated in figure 4 (Mention representation).

Context representation: This representation captures information about the context surrounding the entity mention. Context representation is further divided into two parts, left and right context representation. The left context consists of a sequence of tokens within a sentence from the start of a sentence till the last token of entity mention. The right context consists of a sequence

of tokens from the start of entity mention till the end of a sentence. We use bi-directional LSTM encoders (Graves et al., 2013) to encode token level sequences of both context to a fixed dimensional vector. Formally, for an entity mention $m_{j,k}^i$ present in a sentence s^i , decompose s^i into a sequence of tokens $s_1^i, s_2^i, \dots, s_k^i$ for the left context, and $s_j^i, s_{j+1}^i, \dots, s_{|s^i|}^i$ for the right context, where $|s^i|$ denotes the number of tokens in the sentence. Every token will have a corresponding vector representation in a lookup tables for token. The token sequence is then fed one by one to a bi-directional LSTM encoder, and the final output will be used as feature representation. We denote this whole process by function $F_{lc} : (\mathcal{M}, \mathcal{S}) \rightarrow \mathbb{R}^{D_{lc}}$ for computing left context and $F_{rc} : (\mathcal{M}, \mathcal{S}) \rightarrow \mathbb{R}^{D_{rc}}$ for computing right context. D_{lc} and D_{rc} are the number of dimensions for the left context and the right context representation, respectively. The whole process is illustrated in figure 4 (Left and right context representation).

The context representation described above is slightly different from what was proposed in (Shimaoka et al., 2016), here we include entity mention tokens within both left and right context, to explicitly encode context relative to an entity mention.

In the end, we concatenate entity mention and its context representation into a single D_f dimensional vector, where $D_f = D_m + D_{lc} + D_{rc}$. This complete process is denoted by a function

$F : (\mathcal{M}, \mathcal{S}) \rightarrow \mathbb{R}^{D_f}$ given by:

$$F(m_{j,k}^i, s^i) = F_m(m_{j,k}^i) \oplus F_{lc}(m_{j,k}^i, s^i) \oplus F_{rc}(m_{j,k}^i, s^i) \quad (1)$$

where \oplus denotes vector concatenation. For brevity, we will now omit the use of subscript j, k from $m_{j,k}^i$ and $l_{j,k}^i$, and will use f^i to denote feature representation for entity mention and its context obtained via equation 1.

3.4 Feature and label embeddings

Similar to Yogatama et al. (2015) and Ren et al. (2016), we embed feature representations and labels in a same dimensional space such that an object is embedded closer to the objects that share similar types than the objects that do not. Formally, we are trying to learn linear mapping functions $\phi_{\mathcal{M}} : \mathbb{R}^{D_f} \rightarrow \mathbb{R}^{D_e}$ and $\phi_{\mathcal{L}} : \mathbb{R}^{D_K} \rightarrow \mathbb{R}^{D_e}$, where D_e is the size of embedding space. These mappings are given by:

$$\phi_{\mathcal{M}}(f^i) = f^{iT} U; \quad \phi_{\mathcal{L}}(l_t^i) = l_t^{iT} V \quad (2)$$

where, $U \in \mathbb{R}^{D_f \times D_e}$ and $V \in \mathbb{R}^{D_K \times D_e}$ are projection matrices for features representations and type labels respectively and l_t^i is one-hot vector representation for label t .

We assign a score to each label type t and feature vector as a dot product of their embeddings. Formally, we denote a score as:

$$s(f^i, l_t^i) = \phi_{\mathcal{M}}(f^i) \cdot \phi_{\mathcal{L}}(l_t^i) \quad (3)$$

3.5 Optimization

We use two different loss functions to model *clean* and *noisy* entity mentions. For *clean* entity mentions, we use a hinge loss function. The intuition is simple: maintain a margin, centered at zero, between positive and negative type scores. The scores are computed by similarity between an entity mention and label types (eq. 3). Hinge loss function has two advantages. First, it intuitively separates positive and negative labels during inference. Second, it is independent of data dependent parameter. Formally, for a given entity mention m^i and its label l^i we compute the associated loss as given by:

$$L_c(m^i, l^i) = \sum_{t \in \gamma} \max(0, 1 - s(m^i, l_t^i)) + \sum_{t \in \bar{\gamma}} \max(0, 1 + s(m^i, l_t^i)) \quad (4)$$

where γ and $\bar{\gamma}$ are set of indices that have positive and negative labels respectively.

For *noisy* entity mentions, we propose a variant of a hinge loss where, like L_c , score for all negative labels should go below -1 . However, for positive labels, as we don't know which labels are relevant to entity mention's local context, we propose that the maximum score from the set of given positive labels should be greater than one. This maintains a margin between all negative types and the most relevant positive type. Formally, *noisy* label loss, L_n is defined as:

$$L_n(m^i, l^i) = \sum_{t \in \bar{\gamma}} \max(0, 1 + s(m^i, l_t^i)) + \max(0, 1 - s(m^i, l_{t^*}^i)); \quad t^* = \arg \max_{t \in \gamma} s(m^i, l_t^i) \quad (5)$$

Again, using this loss function makes it intuitive to set a threshold of zero during inference.

These loss functions are different from the loss functions used in (Yogatama et al., 2015; Ren et al., 2016) in a way that, we make strict absolute criteria to distinguish between positive and negative labels. Whereas in (Yogatama et al., 2015; Ren et al., 2016) positive labels should have a higher score than negative labels. As their scoring is relative, the final result varies on the threshold used to separate positive and negative labels.

To train the partitioned dataset together, we formulate the joint objective problem as:

$$\min_{\theta} O = \sum_{m \in \mathcal{M}_c} L_c(m, l) + \sum_{m \in \mathcal{M}_n} L_n(m, l) \quad (6)$$

where θ is the collection of all model parameters that needs to be learned. To jointly optimize the objective O , we use Adam (Kingma and Ba, 2014), a stochastic gradient-based optimization algorithm.

3.6 Inference

For every entity mention in set \mathcal{M} from \mathcal{D}_{test} , we perform a top-down search in the given type hierarchy Ψ , and estimate the correct type path Ψ_* . Starting from the tree root, we recursively compute the best type among node's children by computing its score with obtained feature representations. We select the node that has maximum score among other nodes. We continue this process till a leaf node is encountered or the score associated

with a node falls below an absolute threshold zero. The threshold is fixed across all datasets used.

3.7 Transfer learning

We want to investigate, whether the feature representations learnt for an entity mention are useful. We study what contribution these feature representations make to an existing feature engineering based method such as AFET. We learn the proposed model on one training dataset, namely Wiki dataset, which has the highest number of entity mentions among other datasets and use this model to generate representations that is $F(m_{j,k}^i, s^i)$ for another training and testing data. These representations, which are D_f dimensional vectors, are used as feature for an existing state-of-the-art model, AFET, in place of the hand-crafted features that were originally used. AFET model is then trained using these feature representations. We call this as feature level transfer learning. On the other hand, we also evaluate model level transfer learning, where we initialize weights of LSTM encoders for a new dataset with the weights learnt from the model trained on another dataset, namely Wiki dataset.

4 Experiments

4.1 Datasets used

We evaluate the proposed model on three publicly available datasets, provided in a pre-processed tokenized format by Ren et al. (2016). Statistics of the datasets used in this work are shown in Table 1. The details of the datasets are as follows:

Wiki/FIGER(GOLD): The training data consists of Wikipedia sentences and was automatically generated in distant supervision paradigm, by mapping hyperlinks in Wikipedia articles to Freebase. The test data, mainly consisting of sentences from news reports, was manually annotated as described in (Ling and Weld, 2012).

OntoNotes: OntoNotes dataset consists of sentences from newswire documents present in OntoNotes text corpus (Weischedel et al., 2013). DBpedia spotlight (Daiber et al., 2013) was used to automatically link entity mention in sentences to Freebase. For this corpus, manually annotated test data was shared by Gillick et al. (2014).

BBN: BBN dataset consists of sentences from Wall Street Journal articles and is completely manually annotated (Weischedel and Brunstein, 2005). Please refer to (Ren et al., 2016) for more details

Datasets	Wiki/FIGER(GOLD)	OntoNotes	BBN
# types	128	89	47
# training mentions	2690286	220398	86078
# testing mentions	563	9603	13187
% clean training mentions	64.58	72.61	75.92
% clean testing mentions	88.28	94.00	100
% pronominal testing mentions ¹	0.00	6.78	0.00
Max hierarchy depth	2	3	2

Table 1: Statistics of the datasets used in this work.

of the datasets.

4.2 Evaluation settings

4.2.1 Baselines

We compared the proposed model with state-of-the-art entity classification methods²: (1) **FIGER** (Ling and Weld, 2012); (2) **HYENA** (Yosef et al., 2012); (3) **AFET-NoCo** (Ren et al., 2016): AFET without data based label-label correlation modeled in loss function; (4) **AFET-CoH** (Ren et al., 2016): AFET with hierarchy based label-label correlation modeled in loss function; (5) **AFET** (Ren et al., 2016); (6) **Attentive** (Shimaoka et al., 2016): An attentive neural network based model.

We compare these baselines with variants of our proposed model: (1) **our**: complete model; (2) **our-AIIC** assuming all mentions are *clean*; (3) **our-NoM** without mention representation.

4.2.2 Experimental setup

We use Accuracy or Strict-F1 score, Macro-averaged F1 score, and Micro-averaged F1 score as metrics for evaluation. Existing methods for FETC use same measures (Ling and Weld, 2012; Yogatama et al., 2015; Shimaoka et al., 2016; Ren et al., 2016). We removed entity mentions that do not have any label in training as well as test set. We also remove entity mentions that have spurious indices (i.e entity mention length of 0).³ For all the three datasets, we randomly sampled 10% of the test set, and use it as a development set, on which we tune model parameters. The remaining 90% is used for final evaluation. For all our experiments, we train each model using same hyperparameters five times and report their performance in terms of micro-F1 score on the development set as

¹We considered an entity mention as pronominal, if all of its tokens have POS tag as pronoun.

²Whenever possible, the baselines result are reported from (Ren et al., 2016), otherwise we re-implemented baseline methods based on description available in corresponding papers.

³The code to replicate the work is available at <https://github.com/abhipef/fnet>

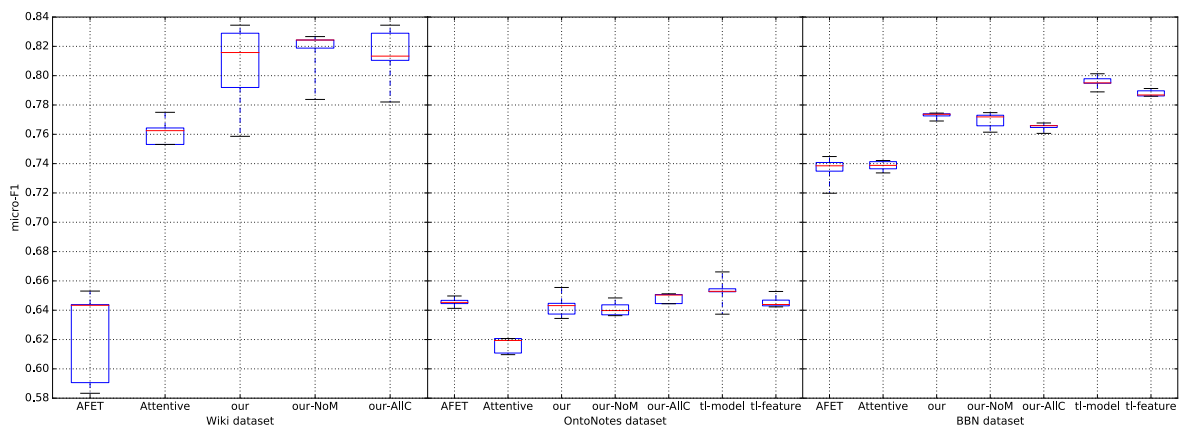


Figure 5: These box-plots show the performance of different baselines on validation set. The red line, boxes and whiskers indicate the median, quartiles and range.

shown in Figure 5. On Wiki dataset, we observed a large variance in performance as compared to other two datasets. This might be because of the fact that Wiki dataset has a very small development set. From each of these five runs, we pick the best performing model based on the development set and report its result on the test set.

Hyperparameter setting: All the neural network based models in this paper used 300 dimensional pre-trained word embeddings distributed by Pennington et al. (2014). The hidden-layer size of word level bi-directional LSTM was 100, and that of character level LSTM was 200. Vectors for character embeddings were randomly initialized and were of size 200. We use dropout with the probability of 0.5 on the output of LSTM encoders. The embedding dimension used was 500. We use Adam (Kingma and Ba, 2014) as optimization method with learning rate of 0.0005-0.001 and mini-batch size in the range of 800 to 1500. The proposed model and some of the baselines were implemented using TensorFlow⁴ framework.

4.3 Transfer learning

In feature level transfer learning, we use the best performing proposed model trained on Wiki dataset to generate representations that is D_f dimensional vector for every entity mention present in the train, development, and test set of the BBN and the OntoNotes dataset. Figure 4 illustrates an example for the encoding process. Then we use these representations as a feature vector in place of the user-defined features and train the AFET

model. Its hyper-parameters were tuned on the development set. These results are shown in table 2 as **feature level transfer-learning**.

In model level transfer learning, we use the learnt weights of LSTM encoders from the best performing proposed model trained on Wiki dataset and initialize the LSTM encoders of the same model with these weights while training on BBN and OntoNotes datasets. These results are shown in table 2 as **model level transfer learning**.

4.4 Performance comparison and analysis

Table 2 shows the results of the proposed method, its variants and the baseline methods.

Comparison with other feature learning methods: The proposed model and its variants (**our-AIIC**, **our-NoM**) perform better than the existing feature learning method by Shimaoka et al. (2016) (**Attentive**), consistently on all datasets. This indicates benefits of the proposed representation scheme and joint learning of representation and label embedding.

Comparison with feature engineering methods: The proposed model performs better than the existing feature engineered methods (**FIGER**, **HYENA**, **AFET-NoCo**, **AFET-CoH**) consistently across all datasets on Micro-F1 and Macro-F1 evaluation metrics. These methods do not model label-label correlation based on data. In comparison with **AFET**, the proposed model outperforms AFET on Wiki and BBN dataset in terms of Micro-F1 evaluation metric. This indicates benefits of feature learning as well as data driven label-label correlation. We do a type-wise perfor-

⁴<http://tensorflow.org/>

Typing methods	Wiki/FiGER(GOLD)			OntoNotes			BBN		
	Acc.	Ma-F1	Mi-F1	Acc.	Ma-F1	Mi-F1	Acc.	Ma-F1	Mi-F1
FIGER [*] (Ling and Weld, 2012)	0.474	0.692	0.655	0.369	0.578	0.516	0.467	0.672	0.612
HYENA [*] (Yosef et al., 2012)	0.288	0.528	0.506	0.249	0.497	0.446	0.523	0.576	0.587
AFET-NoCo [*] (Ren et al., 2016)	0.526	0.693	0.654	0.486	0.652	0.594	0.655	0.711	0.716
AFET-CoH [*] (Ren et al., 2016)	0.433	0.583	0.551	0.521	0.680	0.609	0.657	0.703	0.712
AFET [*] (Ren et al., 2016)	0.533	0.693	0.664	0.551	0.711	0.647	0.670	0.727	0.735
AFET ^{†‡} (Ren et al., 2016)	0.509	0.689	0.653	0.553	0.712	0.646	0.683	0.744	0.747
Attentive [†] (Shimaoka et al., 2016)	0.581	0.780	0.744	0.473	0.655	0.586	0.484	0.732	0.724
our-AIIC [†]	0.662	0.805	0.770	0.514	0.672	0.626	0.655	0.736	0.752
our-NoM [†]	0.646	0.808	0.768	0.521	0.683	0.626	0.615	0.742	0.755
our [†]	0.658	0.812	0.774	0.522	0.685	0.633	0.604	0.741	0.757
model level transfer-learning [†]	-	-	-	0.531	0.684	0.637	0.645	0.784	0.795
feature level transfer-learning [†]	-	-	-	0.471	0.689	0.635	0.733	0.791	0.792

Table 2: Performance analysis of entity classification methods on the three datasets.

mance comparison on OntoNotes dataset in subsection 4.5.

Comparison with variants of our model: The proposed model performs better on all dataset as compared to **our-AIIC** in terms of micro-F1 score. However, we find the performance difference on Wiki and OntoNotes dataset is not statistically significant. We investigated it further and found that across all three datasets, there exist only few entity types for which more than 85% of entity mentions are noisy. These types consist of approximately 3-4% of test set, and our model fails on these types (zero micro-F1 score). However, **our-AIIC** performs relatively well on these types. Examples of such types are: */building*, */person/political_figure*, */GPE/STATE_PROVINCE*. This indicates two limitations of the proposed model. First, the separating of clean and noisy mentions based on the hierarchy has its own inherent limitation of assuming labels within a path are correct. Second, our model learns better if more clean examples are available at the cost of not learning very noisy types. We will try to address these limitations in our future work. Compared with **our-NoM**, the proposed model performs slightly better across all datasets in terms of micro-F1 score.

Feature level transfer learning analysis: We observed 4.5% performance increase in micro-F1 score of AFET on BBN dataset, after replacing hand-crafted features with feature representations generated by the proposed model. This indicates usefulness of the learnt feature representations. However, if we repeat the same process with OntoNotes dataset, there is only a subtle change in performance. This is majorly because of the data distribution of OntoNotes dataset is different from

that of Wiki dataset. This issue is discussed in the next subsection.

Model level transfer learning analysis: In model level transfer learning, sharing knowledge from similar dataset (Wiki to BBN) increases the performance by 3.8% in terms of micro-F1 score. However, sharing knowledge from Wiki to OntoNotes dataset slightly increases the performance by 0.4% in terms of micro-F1 score.

4.5 Case analysis: OntoNotes dataset

We observed three things; (i) all models perform relatively poor on OntoNotes dataset compared to their performance on other two datasets; (ii) the proposed model outperforms other models including AFET on the other two datasets, but gave worse performance on OntoNotes dataset; (iii) the two variants of transfer learning significantly improve performance of the proposed model on the BBN dataset but resulted in only a subtle performance change on OntoNotes dataset.

Statistics of the dataset (Table 1) indicates that presence of pronominal or other kinds of mentions are relatively higher in OntoNotes (6.78% in test set) than the other two datasets (0% in test set). Examples of such mentions are *100 people*, *It*, *the director*, etc. Table 3 shows 20 randomly sampled entity mentions from test set of OntoNotes datasets. Some of these mentions are very generic and likely to be dependent on

^{*}These results are from (Ren et al., 2016) that also uses 10% of the test set as development set and the remaining for evaluation.

[‡]We used the publicly available code distributed by Ren et al. (2016).

[†]All of these results are on exact same train, development and test set.

previous sentences. As all the methods use features solely based on the current sentence, they fail to transfer cross-sentence boundary knowledge. Removing pronominal mentions from test set increases the performance of all feature learning methods by around 3%.

his	thousands of angry people
A reporter	export competitiveness
Freddie Mac	Messrs. Malson and Seelenfreund
the numbers	Hollywood and New York
his explanation	April
volatility	This institution
their hands	the 1987 crash
it	January 4th
Macau	investment enterprises
France	any means

Table 3: 20 randomly sampled entity mentions present in the test set of OntoNotes dataset.

Next we analyse where the proposed model is failing as compared to AFET. For this, we look at type-wise performance for the top-10 most frequent types in the OntoNotes test dataset. Results are shown in Table 4. Compared to AFET, the proposed model performs better in all types except *other* in the top-10 frequent types. The *other* type, which is dominant in test set (42.6% of entity mentions are of type *other*) and is a collection of multiple broad subtypes such as *product*, *event*, *art*, *living_thing*, *food*. Performance of AFET significantly drops (*AFET-NoCo*) when data-driven label-label correlation is ignored, which indicates that modeling data-driven correlation helps. However, as shown in Figure 2a, the use of label-label correlation depends on appropriate values of parameters which vary from one dataset to another.

Label type	Support	our			AFET		
		Prec.	Rec.	F-1	Prec.	Rec.	F-1
/other	42.6%	0.838	0.809	0.823	0.774	0.962	0.858
/organization	11.0%	0.588	0.490	0.534	0.903	0.273	0.419
/person	9.9%	0.559	0.467	0.508	0.669	0.352	0.461
/organization/company	7.8%	0.932	0.166	0.282	1.0	0.127	0.225
/location	7.5%	0.687	0.796	0.737	0.787	0.609	0.687
/organization/government	2.1%	0	0	0	0	0	0
/location/country	2.0%	0.783	0.614	0.688	0.838	0.498	0.625
/other/legal	1.8%	0	0	0	0	0	0
/location/city	1.8%	0.919	0.610	0.733	0.816	0.637	0.715
/person/political_figure	1.6%	0	0	0	0	0	0

Table 4: Performance analysis of the proposed model and AFET on top 10 (in terms of type frequency) types present in OntoNotes dataset.

5 Conclusion and Future Work

In this paper, we propose a neural network based model for the task of fine-grained entity classification. The proposed model learns representations for entity mention, its context and incorporate label noise information in a variant of non-parametric hinge loss function. Experiments show that the proposed model outperforms existing state-of-the-art models on two publicly available datasets without explicitly tuning data dependent parameters.

Our analysis indicates the following observations. First, OntoNotes dataset has a different distribution of entity mentions compared with other two datasets. Second, if data distribution is similar, then transfer learning is very helpful. Third, incorporating data-driven label-label correlation helps in the case of labels of mixed types. Fourth, there is an inherent limitation in assuming all labels to be clean if they belong to the same path of the hierarchy. Fifth, the proposed model fails to learn label types that are very noisy.

Future work could analyse the effect of label noise reduction techniques on the proposed model, revisiting the definition of clean and noisy labels and modeling label-label correlation in a principled way that is not dependent on dataset specific parameters.

Acknowledgments

We thank the anonymous reviewers for their invaluable and insightful comments. Abhishek is supported by MHRD fellowship, Government of India. We acknowledge the use of computing resources made available from the Board of Research in Nuclear Science (BRNS), Dept. of Atomic Energy (DAE), Govt. of India sponsored project (No.2013/13/8-BRNS/10026) by Dr. Aryabartta Sahu at Department of Computer Science and Engineering, IIT Guwahati.

References

- Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Semantic Web Conference*, ISWC’07/ASWC’07, pages 722–735, Berlin, Heidelberg, Springer-Verlag.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A

- collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.
- Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.
- Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86. AAAI Press.
- Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 121–124, New York, NY, USA. ACM.
- Xin Dong, Evgeniy Gabilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmman, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, New York, NY, USA. ACM.
- Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1243–1249. AAAI Press.
- Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.
- Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780, November.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S. Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1891–1901, Doha, Qatar, October. Association for Computational Linguistics.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.
- Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Thomas Lin, Mausam, and Oren Etzioni. 2012. No noun phrase left behind: Detecting and typing un-linkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 893–903, Jeju Island, Korea, July. Association for Computational Linguistics.
- Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 94–100. AAAI Press.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.
- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August. Association for Computational Linguistics.
- T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2302–2310. AAAI Press.
- Lili Mou, Ran Jia, Yan Xu, Ge Li, Lu Zhang, and Zhi Jin. 2016. Distilling word embeddings: An encoding approach. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM '16, pages 1977–1980, New York, NY, USA. ACM.

- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.
- Lorien Y. Pratt. 1993. Discriminability-based transfer between neural networks. In *Advances in Neural Information Processing Systems 5*, pages 204–211, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.
- Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378, Austin, Texas, November. Association for Computational Linguistics.
- Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1057–1067, Cambridge, MA, October. Association for Computational Linguistics.
- Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74, San Diego, CA, June. Association for Computational Linguistics.
- Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, pages 697–706, New York, NY, USA. ACM.
- Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487, Montréal, Canada, June. Association for Computational Linguistics.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1225–1237. IEEE.
- Ralph Weischedel and Ada Brunstein. 2005. BBN Pronoun Coreference and Entity Type Corpus LDC2005T33. *Linguistic Data Consortium, Philadelphia*, 112.
- Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.
- Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 291–296, Beijing, China, July. Association for Computational Linguistics.
- Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters*, pages 1361–1370, Mumbai, India, December. The COLING 2012 Organizing Committee.

Event extraction from Twitter using Non-Parametric Bayesian Mixture Model with Word Embeddings

Deyu Zhou[†] Xuan Zhang[†] Yulan He[§]

[†] School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China

[§] School of Engineering and Applied Science, Aston University, UK
{d.zhou, xuanzhang}@seu.edu.cn, y.he@cantab.net

Abstract

To extract structured representations of newsworthy events from Twitter, unsupervised models typically assume that tweets involving the same named entities and expressed using similar words are likely to belong to the same event. Hence, they group tweets into clusters based on the co-occurrence patterns of named entities and topical keywords. However, there are two main limitations. First, they require the number of events to be known beforehand, which is not realistic in practical applications. Second, they don't recognise that the same named entity might be referred to by multiple mentions and tweets using different mentions would be wrongly assigned to different events. To overcome these limitations, we propose a non-parametric Bayesian mixture model with word embeddings for event extraction, in which the number of events can be inferred automatically and the issue of lexical variations for the same named entity can be dealt with properly. Our model has been evaluated on three datasets with sizes ranging between 2,499 and over 60 million tweets. Experimental results show that our model outperforms the baseline approach on all datasets by 5-8% in F-measure.

1 Introduction

Event extraction from texts is to automatically extract key information of events such as what happened to whom, when and where. Previous research mainly focused on news articles, the best and abundant source of newsworthy events. With the increasing popularity of social media platforms, events are also reported and discussed in

Table 1: An example of several tweets describing the same event about “*Space shuttle Atlantis landed at Kennedy Space Center in Florida on 2011/07/08*”.

Boom! #shuttle #Atlantis is back!

The shuttle is down, welcome back Atlantis, goodbye shuttle program.

Atlantis lands safely in Florida, marking the end of NASA's 30-yr space shuttle programme.

Space shuttle Atlantis lands at Kennedy space center, ending NASA's 30-year shuttle program.

social media apart from news articles. It was reported in (Petrovic et al., 2013) that even 1% of public Twitter stream covers 95% of all events on newswire. Extracting events from social media makes it possible to quickly understand what is being discussed. It can be further integrated into downstream applications such as tracking the public's viewpoints towards a certain event. However, due to the difficulty in acquiring annotated data for training and the short and informal text commonly appeared in social media, traditional approaches (Grishman et al., 2005; Tanev et al., 2008; Piskorski et al., 2008) to event extraction from news articles are no longer applicable in social media data. Nevertheless, one important characteristic of social media data is that for most newsworthy events, there might be a high volume of redundant messages referring to the same event. An example of several tweets describing one event is given in Table 1.

Approaches to event extraction from social media have largely explored the redundancy characteristic (Xia et al., 2015; Popescu et al., 2011; Abdelhaq et al., 2013). Most of the previous methods aim to discover new or previously unidenti-

fied events without extracting structured representations of events. Ritter et al. (2012) presented a system called TwiCal to extract and categorize events from Twitter. The strength of association between each named entity y and date d is measured based on the number of co-occurring tweets in order to form a binary tuple $\langle y, d \rangle$ to represent an event. However, TwiCal relies on a supervised sequence labeler trained on tweets annotated with event mentions for the identification of event-related phrases.

Assuming that each tweet message $m \in \{1..M\}$ is assigned to one event instance e , while e is modeled as a joint distribution over the named entities y , the date/time d when the event occurred, the location l where the event occurred and the event-related keywords k , Zhou et al. (2014; 2015) proposed an unsupervised Bayesian model called latent event model (LEM) for event extraction from Twitter. However, LEM requires the number of events to be known beforehand, which is not realistic in practical applications. To address this limitation, in this paper, a non-parametric mixture model for event extraction is proposed, in which the number of events is inferred automatically from data. Moreover, the lexical variation of the same named entity, for example, “Charles” and “The Prince of Wales”, if identified properly, could be exploited to help in detecting the same event described in tweets with different mentions. To this end, we further extend the non-parametric mixture model to incorporate word embeddings generated using neural language modelling.

The main contributions of the paper are summarized below:

- We propose a non-parametric approach called the Dirichlet Process Event Mixture Model (DPEMM) to extract structured events information. It avoids the problem of pre-setting the number of events, a common issue in latent Dirichlet allocation (LDA) based approaches.
- We extend DPEMM by incorporating word embeddings to deal with the issue of using multiple mentions to refer to the same named entity.
- The proposed approaches have been evaluated on three datasets and a significant improvement on F-measure compared to the baseline approach is observed.

2 Related Work

Research on event extraction of tweets can be divided into domain-specific and open domain approaches. Domain-specific approaches typically focus on one particular type of events. For example, Panem et al. (2014) proposed an algorithm to extract attribute-value pairs and map such pairs to manually generated schemas for natural disaster events. Evaluation was carried out on 58,000 tweets for 20 events and the system can fill such event schemas with an F-measure of 60%. TSum4act (Nguyen et al., 2015) was designed for disaster responses based on tweets and has been evaluated on a dataset containing 230,535 tweets. Anantharam et al. (Anantharam et al., 2014) focused on extracting city events by solving a sequence labeling problem. Evaluation was carried out on a real-world dataset consisting of event reports and tweets collected over four months from San Francisco Bay Area.

Open domain event extraction approaches are not limited to a specific event type or topic. Benson et al. (2011) proposed a structured graphical model which simultaneously analyzed individual messages, clustered, and induced a canonical value for each event. Popescu et al. (2011) focused on detecting events involving known entities from Twitter. Experimental results showed that events centered on specific entities can be extracted with 70% precision and 64% recall. Liu et al. (2012) worked on social events extraction for social network construction using a factor graph by harvesting the redundancy in tweets. Experiments were conducted on manually annotated data set and results showed that it achieved a gain of 21% in F-measure. In (Abdelhaq et al., 2013), a system called EvenTweet was constructed to extract localized events from a stream of tweets in real-time. The extracted events are described by start time, location and a number of related keywords. Armengo et al. (2015) proposed a model named Tweet-SCAN based on hierarchical Dirichlet process to detect events from geo-located tweets. To extract more information, a system called SEEFT (Wang et al., 2015) used links in tweets and combined tweets and linked articles to identify events. Xia et al. (2015) proposed a framework combining text, image and geo-location information to detect events with low spatial and temporal deviation.

Our proposed method belongs to the open do-

main category. Different from the previous methods, our model can automatically identify the number of events in the corpus and deal with lexical variations of named entities using word embeddings generating from neural language modelling.

3 Methodology

Our proposed model for event extraction is based on a typical non-parametric mixture model, Dirichlet Process Mixture Model (DPMM) (Green and Richardson, 2001; Ishwaran and Zarepour, 2002) in which the number of active clusters is automatically learned from the data. We first give a brief introduction to DPMM. In DPMM, observation x_i is assumed to be derived from the following model:

$$\begin{aligned}\pi|\alpha &\sim \text{Dirichlet}(\alpha/K, \dots, \alpha/K) \\ c_i|\pi &\sim \text{Multinomial}(\pi) \\ \phi_k|G_0 &\sim G_0 \\ x_i|c_i, \{\phi_k\}_{k=1}^K &\sim F(\phi_{c_i})\end{aligned}$$

where K denoting the number of components in the mixture model and can go to infinity, π is the mixture weights of each component, ϕ_k is the parameter of the k th component, c_i denotes the index of components, $F(\phi_{c_i})$ denotes the distribution of x_i with parameter ϕ_{c_i} . In this model, π can be generated by stick-breaking model (Pitman, 2002) and Chinese restaurant process (Aldous, 1985).

Suppose that all the observations are generated by DPMM and the variable of observation x_i is θ_i , which has the following conditional distribution:

$$\theta_i|\theta_1, \dots, \theta_{i-1} \sim \sum_{k=1}^K \frac{n_k}{i-1+\alpha} \delta_{\phi_k} + \frac{\alpha}{i-1+\alpha} G_0$$

where ϕ_1, \dots, ϕ_k are the distinct values of θ , n_k is the number of observations that belong to component k , δ_{ϕ_k} is a probability measure concentrated on ϕ_k , which returns 1 when $\theta_i = \phi_k$, G_0 is the base probability measure and generates new ϕ with probability $\frac{\alpha}{i-1+\alpha}$.

3.1 Dirichlet Process Event Mixture Model (DPEMM)

We propose a Dirichlet Process Event Mixture Model (DPEMM) in which each event is represented as a 4-tuple $\langle y, l, k, d \rangle$, where y stands for non-location named entity, l for location, k for

event-related keyword and d for date. It is worth noting that y, l, k is not atomic and could be a set by itself. One event can have multiple named entities, locations or keywords. Also, some elements of the 4-tuple might be absent if no associated information can be found in tweets. Assuming that the data contains an infinite number of events and each event is modeled as a joint distribution over y, l, k and d , the model can be viewed as a Bayesian mixture model.

The generative process of the proposed model is given below.

- Draw event distribution $\pi \sim \text{Dirichlet}(\frac{\alpha}{K}, \dots, \frac{\alpha}{K})$.
- For each event e , draw multinomial distribution $\theta_e \sim \text{Dirichlet}(\beta)$, $\psi_e \sim \text{Dirichlet}(\eta)$, $\omega_e \sim \text{Dirichlet}(\lambda)$, $\phi_e \sim \text{Dirichlet}(\gamma)$.
- For each tweet \mathbf{t} :
 - Draw an event from event distribution $e \sim \text{Multinomial}(\pi)$.
 - For each non-location named entity occurred in \mathbf{t} , choose a named entity $y \sim \text{Multinomial}(\theta_e)$.
 - For each location occurred in \mathbf{t} , choose a location $l \sim \text{Multinomial}(\psi_e)$.
 - For each keyword occurred in \mathbf{t} , choose a keyword $k \sim \text{Multinomial}(\omega_e)$.
 - For each date occurred in \mathbf{t} , choose a date $d \sim \text{Multinomial}(\phi_e)$.

Here, K is the number of events and can go to infinity. To estimate the parameters of the model, we employ Markov chain sampling methods (Neal, 2000). As K goes to infinity, we cannot represent the infinite number of $\theta_e, \psi_e, \omega_e$ and ϕ_e explicitly. Therefore, we perform Gibbs sampling for only those parameters that are currently associated with some observations. Gibbs sampling for the event label e_i of tweet i is based on the following conditional probabilities:

If e_i is assigned with a previously seen event e ,

$$\begin{aligned}P(e_i = e|e_{-i}, s_i, t_{-i}) &= b \frac{n_e^{-i}}{n-1+\alpha} \\ &\prod_{y \in y_i} \int F_y(\theta_e) dH_y(\theta_e) \prod_{l \in l_i} \int F_l(\psi_e) dH_l(\psi_e) \\ &\prod_{k \in k_i} \int F_k(\omega_e) dH_k(\omega_e) \prod_{d \in d_i} \int F_d(\phi_e) dH_d(\phi_e)\end{aligned}$$

If e_i is assigned with a new event,

$$P(e_i = e_{new} | e_{-i}, s_i, d_i, t_{-i}) = b \frac{\alpha}{n-1+\alpha} \prod_{y \in y_i} \int F_y(\theta) dG_0(\theta) \prod_{l \in l_i} \int F_l(\psi) dG_0(\psi) \prod_{k \in k_i} \int F_k(\omega) dG_0(\omega) \prod_{d \in d_i} \int F_d(\phi) dG_0(\phi)$$

, where b is the normalizing constant that makes the probabilities sum to 1, e_{-i} is the event assignment of all the other tweets excluding the data from i th tweet, s_i is the four-tuple $\langle y_i, l_i, k_i, d_i \rangle$, n is the total number of tweets, n_e^{-i} is the number of tweets assigned with event label e excluding the current assignment, $F_y(\theta_e)$ is the multinomial distribution over non-location named entities with prior θ_e , $F_l(\psi_e)$ over locations with ψ_e , $F_k(\omega_e)$ over keywords with ω_e , and $F_d(\phi_e)$ over dates with ϕ_e . $H_y(\theta_e)$ is the posterior distribution of parameters based on the prior $G_0(\theta_e) \sim \text{Dirichlet}(\beta)$ and all observations y_j for which $j \neq i$ and $e_j = e$, and similarly for $H_l(\psi_e)$, $H_k(\omega_e)$ and $H_d(\phi_e)$.

We then derive the following formulae:

If e_i is assigned with a previously seen event e ,

$$P(e_i = e | e_{-i}, s_i, t_{-i}) = b \frac{n_e^{-i}}{n-1+\alpha} \prod_{y \in y_i} \frac{n_{e,y}^{-i} + \beta}{\sum_{t=1}^Y (n_{e,y,t}^{-i} + \beta)} \prod_{l \in l_i} \frac{n_{e,l}^{-i} + \eta}{\sum_{t=1}^L (n_{e,l,t}^{-i} + \eta)} \prod_{k \in k_i} \frac{n_{e,k}^{-i} + \lambda}{\sum_{t=1}^K (n_{e,k,t}^{-i} + \lambda)} \prod_{d \in d_i} \frac{n_{e,d}^{-i} + \gamma}{\sum_{t=1}^D (n_{e,d,t}^{-i} + \gamma)}$$

If e_i is assigned with a new event e' ,

$$P(e_i = e' | e_{-i}, s_i, t_{-i}) = b \frac{\alpha}{n-1+\alpha} \prod_{y \in y_i} \frac{1}{Y} \prod_{l \in l_i} \frac{1}{L} \prod_{k \in k_i} \frac{1}{K} \prod_{d \in d_i} \frac{1}{D}$$

, where the superscript $-i$ denotes a count excluding data from i th tweet, $n_{e,y}^{-i}$, $n_{e,l}^{-i}$, $n_{e,k}^{-i}$, and $n_{e,d}^{-i}$ denotes the occurrence count of non-location y , location l , keyword k and date d in event e , respectively. t_{-i} denotes all other tweets. $\beta, \eta, \lambda, \gamma$ are the hyperparameters and are set to the same value 1 in the experiments in the paper.

3.2 DPEMM With Word Embeddings

In the proposed model described above, each distinct word is treated separately without consider-

ing their semantic relations. However, the knowledge of semantic relations of words might be useful for event extraction. For example, ‘‘Putin’’ and ‘‘The President of Russia’’ are two different mentions referring to the same person. Knowing such knowledge would help to cluster the following two tweets together, ‘‘*President of Russia attended the opening ceremony of the 119th session of the International Olympic Committee.*’’ and ‘‘*Putin took part in the presentation of Sochi, at the 119th of the IOC.*’’, and hence identify a single event. Moreover, there might exist partitive relations between two location names. For example, Croydon is a part of London. The information will help to identify the same event described as happened in Croydon and London and subsequently improve the accuracy of event extraction.

To incorporate such information about semantic relations between words, we propose another model by employing word embeddings to describe the semantic relations among y or l , which is called DPEMM-WE. Word embedding for each word is often represented in a vector form. In the embedded hyperspace, words that are more semantically or syntactically similar to each other are located closer. We use neural language modeling (Collobert et al., 2011) to learn word representations by discriminating the legitimate phrase from incorrect phrases. Given a sequence of words $p = (w_1, w_2, \dots, w_d)$ with window size d , the goal of the model is to discriminate the sequence of words p (the correct phrase) from a random sequence of words p^r . Thus, the objective function of the model is to minimize the ranking loss with respect to parameters θ :

$$\sum_{p \in \mathfrak{p}} \sum_{r \in \mathfrak{R}} \max(0, 1 - f_\theta(p) + f_\theta(p^r)) \quad (1)$$

, where \mathfrak{p} is the set of all possible text sequences with d words coming from the corpus U , \mathfrak{R} is the dictionary of words, p^r denotes the window of words obtained by replacing the central word of p by the word r and $f_\theta(p)$ is the score of p . The dataset for learning the language model can be constructed by considering all the word sequences in the corpus. Positive examples are the word sequences from the corpus, while negative examples are the same word sequence with the central word replaced by a random one.

Different from DPEMM, in DPEMM-WE, non-location named entities y and locations l are as-

sumed to follow Gaussian distribution to incorporate word embeddings and their prior distributions are assumed to follow Normal-Inverse-Wishart (NIW) distribution, which is conjugated with Gaussian distribution. The probability density function is

$$\begin{aligned} NIW(\mu, \Sigma | \mu_0, \lambda, \Psi, \nu) \\ = \mathcal{N}(\mu | \mu_0, \frac{1}{\lambda} \Sigma) \mathcal{W}^{-1}(\Sigma | \Psi, \nu) \end{aligned}$$

$$\mathcal{N}(\mu | \mu_0, \frac{1}{\lambda} \Sigma) = \frac{e^{-\frac{1}{2}(\mu - \mu_0)^T (\Sigma / \lambda)^{-1} (\mu - \mu_0)}}{\sqrt{|2\pi \Sigma / \lambda|}}$$

$$\mathcal{W}^{-1}(\Sigma | \Psi, \nu) = \frac{|\Sigma|^{\frac{\nu}{2}}}{2^{\frac{\nu p}{2}} \Gamma_p(\frac{\nu}{2})} |\Sigma|^{-\frac{\nu+p+1}{2}} e^{-\frac{1}{2} \text{tr}(\Psi \Sigma^{-1})}$$

where Σ and Ψ are $p \times p$ positive definite matrices and $\Gamma_p(\cdot)$ is the multivariate gamma function. The graphical model of DPEMM-WE is shown in Figure 1.

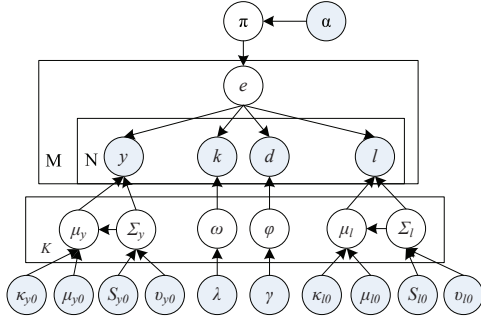


Figure 1: Plate notation of the graphical model DPEMM-WE.

The generative process of DPEMM-WE is given below.

- Draw event distribution $\pi \sim \text{Dirichlet}(\frac{\alpha}{K}, \dots, \frac{\alpha}{K})$.
- For each event, draw Gaussian distribution $\theta_e \sim \text{NIW}(\beta)$, $\psi_e \sim \text{NIW}(\eta)$; draw Multinomial distribution $\omega_e \sim \text{Dirichlet}(\lambda)$, $\phi_e \sim \text{Dirichlet}(\gamma)$.
- For each tweet \mathbf{t} :
 - Draw an event from event distribution $e \sim \text{Multinomial}(\pi)$.
 - For each named entity occurred in \mathbf{t} , choose a named entity $y \sim \text{Gaussian}(\theta_e)$.

- For each location occurred in \mathbf{t} , choose a location $l \sim \text{Gaussian}(\psi_e)$.
- For each keyword occurred in \mathbf{t} , choose a keyword $k \sim \text{Multinomial}(\omega_e)$.
- For each date occurred in \mathbf{t} , choose a date $d \sim \text{Multinomial}(\phi_e)$.

, where $\beta = (\kappa_{y0}, \mu_{y0}, \nu_{y0}, S_{y0})$, $\theta_e = (\mu_y, \Sigma_y)$, $\eta = (\kappa_{l0}, \mu_{l0}, \nu_{l0}, S_{l0})$, $\psi_e = (\mu_l, \Sigma_l)$.

Similar to DPEMM, parameters of the model can be estimated by Gibbs sampling. The sampling equation is given as below:

If e_i is assigned with a previously seen event e ,

$$\begin{aligned} P(e_i = e | e_{-i}, s_i, t_{-i}) &= b \frac{n_e^{-i}}{n - 1 + \alpha} \\ &\prod_{y \in y_i} \int p(y | \theta) p(\theta | \nu_{e,y0}, \kappa_{e,y0}, \mu_{e,y0}, S_{e,y0}, t_{-i}) d\theta \\ &\prod_{l \in l_i} \int p(l | \psi) p(\psi | \nu_{e,l0}, \kappa_{e,l0}, \mu_{e,l0}, S_{e,l0}, t_{-i}) d\psi \\ &\prod_{k \in k_i} \int p(k | \omega) p(\omega | \lambda, t_{-i}) d\omega \\ &\prod_{d \in d_i} \int p(d | \phi) p(\phi | \gamma, t_{-i}) d\phi \end{aligned}$$

If e_i is assigned with a new event e' ,

$$\begin{aligned} P(e_i = e' | e_{-i}, s_i, t_{-i}) &= b \frac{\alpha}{n - 1 + \alpha} \\ &\prod_{y \in y_i} \int p(y | \theta) p(\theta | \nu_{e',y0}, \kappa_{e',y0}, \mu_{e',y0}, S_{e',y0}) d\theta \\ &\prod_{l \in l_i} \int p(l | \psi) p(\psi | \nu_{e',l0}, \kappa_{e',l0}, \mu_{e',l0}, S_{e',l0}) d\psi \\ &\prod_{k \in k_i} \int p(k | \omega) p(\omega | \lambda) d\omega \prod_{d \in d_i} \int p(d | \phi) p(\phi | \gamma) d\phi \end{aligned}$$

, where θ and ψ denote parameter (μ, Σ) .

As

$$\begin{aligned} \int \mathcal{N}(x | \theta) NIW(\theta | \nu, \kappa, \mu, S) d\theta = \\ \mathcal{F}(\nu - D + 1, \mu, \frac{S(\kappa + 1)}{\kappa(\nu - D + 1)}) \end{aligned}$$

the parameters of entities' \mathcal{F} distribution are

given as:

$$\begin{aligned}
\kappa_{e,y} &= \kappa_{y0} + N_e \\
\nu_{e,y} &= \nu_{y0} + N_e \\
\mu_{e,y} &= \frac{\kappa_{y0}\mu_{y0} + N_e\bar{v}_{e,y}}{\kappa_{e,y}} \\
S_{e,y} &= S_{e0} + C_{e,y} \\
&\quad + \frac{\kappa_{e0}N_e}{\kappa_{e,y}}(\bar{v}_{e,y} - \mu_{e0})(\bar{v}_{e,y} - \mu_{e0})^T \\
\bar{v}_{e,y} &= \frac{\sum_{y \in e} v_y}{N_e} \\
C_{e,y} &= \sum_{y \in e} (v_y - \bar{v}_{e,y})(v_y - \bar{v}_{e,y})^T
\end{aligned}$$

, where v_y means the word embedding of entity y . The parameters of locations' \mathcal{T} distribution can be calculated similarly.

3.3 Post-Processing

DPEMM or DPEMM-WE essentially outputs tweet clusters where each cluster represents one event. To further extract structured representation of an event, such as named entities, locations, dates and keywords, from each cluster, we simultaneously look into the probabilities of each event element returned by our models and their co-occurrence frequencies. We assumed that non-location named entities were the most important since an event is usually operated by somebody or something. If an event happened in someplace like "A bomb attack was happened in London", the location is the most important. Therefore, we first select the top 3 non-location named entities ranked by the probability θ_e . For each non-location named entity y , its occurrence frequency needs to exceed T_y . If no such entities exist, the top 3 locations ranked by the probability ψ_e are chosen; otherwise, the location l is chosen based on its co-occurrences with the selected non-location named entities. After that, keywords k are chosen among the top 10 ω_e . Only those keywords with correlation coefficients with the chosen named entities and locations exceeding T_c are selected. Then date d is chosen in a similar way. Here, we define the correlation coefficient between a and b as $Corr(a, b) = \log \frac{\#(a,b)}{\#(b)}$, where $\#(a, b)$ denotes the co-occurrence count of a and b in the same tweet within a tweet cluster and $\#(b)$ denotes the occurrence count of b in all tweets within a tweet cluster. In our experiments, we set the thresholds $T_y = 0.2, T_c = 0.4$.

If the entity or location is in the form of word embeddings, its occurrence frequency is calculated as the occurrence frequencies of all the neighboring words which have cosine similarity values greater than 0.85. The rationale behind our post-processing step is that although tweets have been filtered in the pre-processing step, tweet clusters generated by the proposed models still contain noisy event elements. As such, we select event elements from tweet clusters not only based on their probability distributions given by the proposed models but also taking into account their co-occurrences in each tweet cluster.

4 Experiments

We evaluate the proposed models on three datasets. Dataset I is the First Story Detection (FSD) dataset (Petrovic et al., 2013) containing 2,499 tweets manually annotated with 27 events. These tweets were published between 7th July and 12th September 2011, covering a range of categories such as accidents and science discoveries. Considering that events mentioned in a very few tweets are less likely to be significant, we remove events mentioned in less than 15 tweets and are left with 2,453 tweets annotated with 20 events. Dataset II and III were collected from tweets published in the month of December in 2010 using the Twitter streaming API. Dataset II consists of 6,297 tweets manually annotated with 73 events. All the annotated events in Dataset II are mentioned in at least 15 tweets. Dataset III contains 60 millions unlabelled tweets. We chose LEM (Zhou et al., 2014), the state-of-art approach based on Bayesian modelling for event extraction, as the baseline to compare with the proposed model. For all datasets, pre-processing is done as described in *baseline* (Zhou et al., 2014). A named entity tagger¹ specifically built for Twitter is used for extracting named entities including locations from tweets. A Twitter Part-of-Speech tagger (Gimpel et al., 2011) is used for POS tagging and only words tagged with nouns, verbs or adjectives are kept as candidate keywords. Word embeddings are trained on Dataset III (60 million tweets) using Word2Vec². In this model, a word is used as an input to a log-linear classifier with continuous projection layer and the objective is to predict its neighboring words.

¹<http://github.com/aritter/twitter-nlp>

²<http://code.google.com/p/word2vec/>

We train DPEMM, DPEMM-WE and LEM on an IBM 3850 X5 Linux server equipped with 1.86 Ghz processor and 8 GB DDR3 RAM. The number of Gibbs sampling iterations is set to 1,000 for LEM for all the datasets. For DPEMM, it converges in 16 iterations on Dataset I and 20 iterations on Dataset II and III. While for DPEMM-WE, it converges in 20 iterations on both Dataset II and III.

4.1 Experimental Results

To evaluate the performance of the proposed approaches, we calculate *precision*, *recall*, and *F-measure* on Dataset I and II and only *precision* on Dataset III since it is hard to know exactly how many events are mentioned in such a large dataset. The *precision* is defined based on the following criteria: 1) Do the entity y , location l and the date d refer to the same event? 2) Are the keywords k in accord with the event that other extracted elements y , l , d refer to and are they informative enough to tell us what happened? If the extracted events does not have any keyword, such events are considered as incorrect.

The performance comparison of event extraction results is presented in Table 2. It can be observed that the proposed DPEMM achieves better performance on all the three datasets compared to the baseline approach, with the improvement in F-measure being 6.1% and 7.7% on Dataset I and II, respectively. After incorporating word embeddings into DPEMM, the proposed DPEMM-WE further improves upon DPEMM slightly by 1.45% in F-measure on Dataset II, but more significantly by 4.16% in precision on Dataset III. It verifies our hypothesis that the knowledge about the semantic relations of entities and locations could potentially improve the performance of event extraction. We also compared the proposed models with K-means on Dataset I to justify whether these proposed generative models are better than traditional clustering methods based on co-occurrence. The feature set was constructed by organizing the words in four categories such as y , l , k , d and concatenating the four one-hot feature sets together.

It is worth noting that we did not apply DPEMM-WE on Dataset I because this dataset is very small, consisting of less than 2500 tweets. It is thus unreliable to learn word embeddings from such a small dataset. It is also hard to pre-train word embedding from extra dataset like Wikipedia

Table 2: Comparison of the performance of event extraction on the three datasets.

Dataset I			
Method	Precision(%)	Recall(%)	F-measure(%)
K-means	91.23	55.40	68.93
LEM	79.17	85.00	81.98
DPEMM	86.21	90.00	88.06
Dataset II			
Method	Precision(%)	Recall(%)	F-measure(%)
LEM	62.35	68.49	65.28
DPEMM	70.80	75.34	73.00
DPEMM-WE	71.15	78.08	74.45
Dataset III			
Method	Precision(%)	Number of correctly Events	
LEM	68.25	215	
DPEMM	68.60	342	
DPEMM-WE	72.76	353	

corpus for Dataset I because some words in social media are informal and some words were only mentioned in some specific time slots such as “Dream Act”. Also, word embeddings learned from Dataset III are not beneficial for event extraction in Dataset I since tweets collected in these two datasets were in different periods and a large number of words in Dataset I cannot be found in Dataset III. For example, more than 20% named entities in Dataset I can not be found in the word vocabulary constructed based on Dataset III.

Examples of events extracted by DPEMM and DPEMM-WE are shown in Table 3. It can be observed that the extracted results from DPEMM-WE contain more detailed and accurate information describing the events. For example, for the first event, DPEMM-WE is able to extraction the location information while DPEMM failed to do so. For the third event, DPEMM-WE gives more accurate location information compared to DPEMM. It might attribute to the advantage of incorporating word embeddings which are able to map semantically similar words into nearby locations in the embedding hyperspace. As such, although two tweets might contain different mentions of named entities and locations, they might still be clustered together if these named entities or locations have similar word embeddings.

We observed that the precision achieved by DPEMM on Dataset I is significantly better than LEM on Dataset I and II while similar on Dataset III. We found that DPEMM tended to generate many but smaller clusters compared to LEM. As dataset III is huge, DPEMM might generate some small clusters which do not contain enough information to describe a correct event.

Table 3: Examples of extracted events based on DPEMM and DPEMM-WE.

Event	Method	Entities	Locations	Keywords	Date
1	DPEMM	Biden	-	inevit marriage gay say	2010-12-24
	DPEMM-WE	Biden, Obama	WhiteHouse	marrige gay inevit say	2010-12-24
2	DPEMM	Charles	London	car protest attack contain	2010-12-09
	DPEMM-WE	Charles, Camilla	London, UK	protest car demonstrators attack	2010-12-09
3	DPEMM	-	London	snow close airport ice	2010-12-18
	DPEMM-WE	-	Europ, London, Gatwick	snow delay runaway airport	2010-12-18
4	DPEMM	DreamAct, Reid	-	pass bill vote will	2010-12-09
	DPEMM-WE	DreamAct, Harry, Reid	Senate	vote pass debate bill	2010-12-09
5	DPEMM	WorldCup	Russia	announce will host chose	2010-12-03
	DPEMM-WE	WorldCup, FIFA	Russia, Qatar	news host win will	2010-12-03

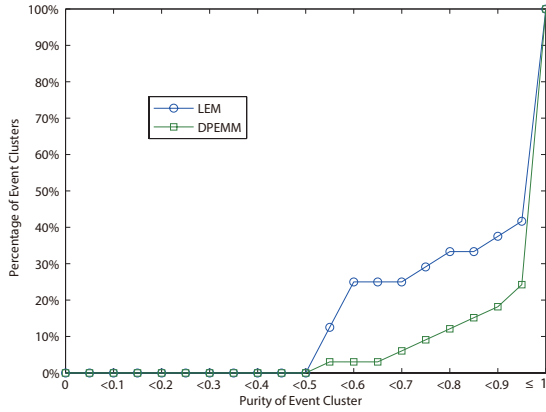


Figure 2: Purities of the clusters on Dataset I.

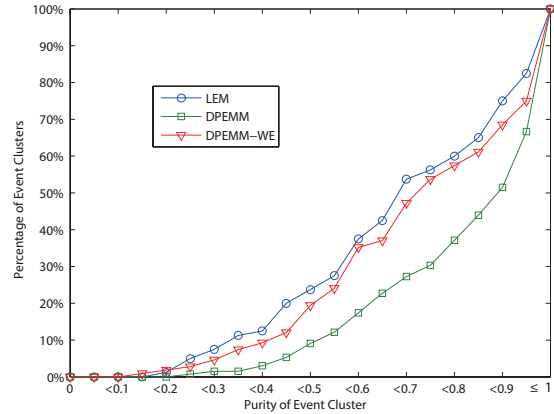


Figure 3: Purities of the clusters on Dataset II.

4.2 Quality of Clusters

As the proposed approaches essentially group tweets into different clusters with each cluster corresponding to an event, we conduct experiments to explore the quality of clusters by a measure of purity, which is defined as $P_e = \frac{n_e}{n}$, where n_e denotes the number of tweets describing the event e extracted from a cluster and n denotes the total number of tweets in the cluster. Since it is difficult to calculate the purity on Dataset III, we only report the results on Dataset I and Dataset II as shown in Figure 2 and 3 respectively.

Each point (x, y) in the figures denotes the percentage y of the clusters whose purity is less than x . Obviously, if the curve is steeper, it means that the percentage of the clusters with low purity is smaller and the quality of the clusters is better. It can be observed that DPEMM achieves the best quality of cluster on both Dataset I and Dataset II, whose precision is lower than DPEMM-WE. Specifically, on Dataset I, more than 80% of clusters generated by DPEMM has the purity value greater than 0.9, compared to only 70% in LEM. It might be attributed to the property of DPEMM

that the cluster is generated dynamically without a preset number of clusters. On Dataset II, both DPEMM and DPEMM-WE achieve better clustering results compared to LEM. However, the purity of clusters generated by DPEMM is slightly higher than that generated by DPEMM-WE. This is somewhat contrary to our prior belief. By further analyzing the results, we found that as more tweets are clustered together using DPEMM-WE, more noisy information such as some named entities with similar word embeddings which are not related to the events is introduced. We present an example of the tweet clusters describing the same event generated by DPEMM and DPEMM-WE in Figure 4. For each method, we use a histogram to indicate the number of tweets which share the same event elements. Regions highlighted in dark or light red colors indicate that the corresponding tweets are event-related. Regions highlighted in blue denote the corresponding tweets are not event-related. It can be observed that the purity of the cluster generated by DPEMM is 91% which is better than DPEMM-WE's 63%. However, the size of the cluster returned by DPEMM is smaller and it failed to extract the location information.

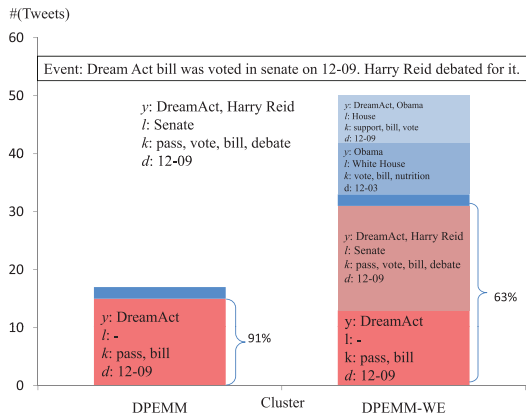


Figure 4: Example tweet clustering results generated by DPEMM and DPEMM-WE.

On the contrary, DPEMM-WE generated a larger cluster and for some tweets, it successfully extracted the location “Senate”. However, more spurious tweets are included because “Harry Reid” is close to both “DreamAct” and “Obama”, and “White House” is close to “Senate” in the word embedding space. Therefore, although DPEMM-WE gives better extraction results overall compared to DPEMM as shown in Table 2, it returns lower purity results because of some noisy information introduced through word embeddings.

5 Conclusions and Future Work

In this paper, we have proposed a model based on the Dirichlet Process mixture model to extract structured event information from social media data. Different from previous approaches for event extraction which require setting the number of events beforehand, it can infer the number of events automatically from data. It is specifically appealing for processing large-scale social media data. Moreover, considering different mentions of names could refer to the same person (and similarly for other named entities such as location), we have proposed to incorporate word embeddings into DPEMM so as to more effectively capture semantically similar words. Experiments have been conducted on three datasets and the proposed approaches achieve better performance on all the datasets in comparison with the baseline approach. In the future, we plan to investigate more effective way in reducing the noise introduced by word embeddings and incorporate emotion information into the proposed models to simultaneously ex-

tract public opinions of the extracted event.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was funded by the National Natural Foundation of China (61528302), the Natural Science Foundation of Jiangsu Province of China (BK20161430), the National Key Research and Development Program of China (2016YFC1306704) and the Collaborative Innovation Center of Wireless Communications Technology.

References

- Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. Eventtweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, pages 1326–1329.
- David J. Aldous. 1985. *Exchangeability and related topics*. Springer.
- Pramod Anantharam, Payam Barnaghi, T. K. Prasad, and Amit P. Sheth. 2014. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology*, 9(10):e110206.
- Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 389–398, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Joan Capdevila, Jess Cerquides, Jordi Nin, and Jordi Torres. 2015. Tweet-scan: An event discovery technique for geo-located tweets. In *Artificial Intelligence Research and Development: Proceedings of the 18th International Conference of the Catalan Association for Artificial Intelligence*, volume 277, pages 110–119.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.
- Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT ’11, pages 42–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

- Peter J. Green and Sylvia Richardson. 2001. Modelling heterogeneity with and without the dirichlet process. *Scandinavian journal of statistics*, 28(2):355–375.
- Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu’s english ace 2005 system description. In *ACE 05 Evaluation Workshop*.
- Hemant Ishwaran and Mahmoud Zarepour. 2002. Exact and approximate sum representations for the dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283.
- Xiaohua Liu, Xiangyang Zhou, Zhongyang Fu, Furu Wei, and Ming Zhou. 2012. Extracting social events for tweets using a factor graph. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1692–1698.
- Radford M. Neal. 2000. Markov chain sampling methods for dirichlet process mixture models. *Computational and Graphical Statistics*.
- Minh-Tien Nguyen, Asanobu Kitamoto, and Tri-Thanh Nguyen. 2015. Tsum4act: A framework for retrieving and summarizing actionable tweets during a disaster for reaction. In *Advances in Knowledge Discovery and Data Mining*, pages 64–75. Springer.
- Sandeep Panem, Manish Gupta, and Vasudeva Varma. 2014. Structured information extraction from natural disaster events on twitter. In *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning*, WebKR ’14, pages 1–8, New York, NY, USA. ACM.
- Saša Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. 2013. Can twitter replace newswire for breaking news? In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*.
- Jakub Piskorski, Hristo Tanev, Martin Atkinson, and Erik Van Der Goot. 2008. Cluster-centric approach to news event extraction. In *International Conference on New Trends in Multimedia and Network Information Systems*, pages 276–290.
- Jim Pitman. 2002. Poisson–dirichlet and gem invariant distributions for split-and-merge transformations of an interval partition. *Combinatorics, Probability & Computing*, 11(05):501–514.
- Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. 2011. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World Wide Web (WWW)*, pages 105–106.
- Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’12, pages 1104–1112, New York, NY, USA. ACM.
- Hristo Tanev, Jakub Piskorski, and Martin Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In *13th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 207–218.
- Yu Wang, David Fink, and Eugene Agichtein. 2015. Seeft: Planned social event discovery and attribute extraction by fusing twitter and web content. In *Ninth International AAAI Conference on Web and Social Media*.
- Chaolun Xia, Jun Hu, Yan Zhu, and Mor Naaman. 2015. What is new in our city? a framework for event extraction using social media posts. In *Advances in Knowledge Discovery and Data Mining*, pages 16–32. Springer.
- Deyu Zhou, Liangyu Chen, and Yulan He. 2014. A simple bayesian modelling approach to event extraction from twitter. In *Proceedings of the The 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 700–705.
- Deyu Zhou, Liangyu Chen, and Yulan He. 2015. An unsupervised framework of exploring events on twitter: Filtering, extraction and categorisation. In *Proceedings of the 29th AAAI Conference (AAAI)*, pages 2468–2474.

End-to-end Relation Extraction using Neural Networks and Markov Logic Networks

Sachin Pawar^{1,2}, Pushpak Bhattacharyya², and Girish K. Palshikar¹

¹TCS Research, Tata Consultancy Services, Pune

²Indian Institute of Technology Bombay, Mumbai

{sachin7.p, gk.palshikar}@tcs.com

pb@cse.iitb.ac.in

Abstract

End-to-end relation extraction refers to identifying boundaries of entity mentions, entity types of these mentions and appropriate semantic relation for each pair of mentions. Traditionally, separate predictive models were trained for each of these tasks and were used in a “pipeline” fashion where output of one model is fed as input to another. But it was observed that addressing some of these tasks jointly results in better performance. We propose a single, joint neural network based model to carry out all the three tasks of boundary identification, entity type classification and relation type classification. This model is referred to as “All Word Pairs” model (AWP-NN) as it assigns an appropriate label to each word pair in a given sentence for performing end-to-end relation extraction. We also propose to refine output of the AWP-NN model by using inference in Markov Logic Networks (MLN) so that additional domain knowledge can be effectively incorporated. We demonstrate effectiveness of our approach by achieving better end-to-end relation extraction performance than all 4 previous joint modelling approaches, on the standard dataset of ACE 2004.

1 Introduction

The task of relation extraction (RE) deals with identifying whether any pre-defined *semantic relation* holds between a pair of *entity mentions* in the given sentence. Pure relation extraction techniques (Zhou et al., 2005; Jiang and Zhai, 2007; Bunescu and Mooney, 2005; Qian et al., 2008) assume that for a sentence, gold-standard entity

mentions (i.e. boundaries as well as types) in it are known. In contrast, end-to-end relation extraction deals with plain sentences without assuming any knowledge of entity mentions in them. The task of end-to-end relation extraction consists of three sub-tasks: i) identifying boundaries of entity mentions, ii) identifying entity types of these mentions and iii) identifying appropriate semantic relation for each pair of mentions. First two sub-tasks correspond to the Entity Detection and Tracking task defined by the the Automatic Content Extraction (ACE) program (Doddington et al., 2004) and the third sub-task corresponds to the Relation Detection and Characterization (RDC) task. ACE standard defined 7 entity types¹: PER (person), ORG (organization), LOC (location), GPE (geopolitical entity), FAC (facility), VEH (vehicle) and WEA (weapon). It also defined 7 coarse level relation types²: EMP-ORG (employment), PER-SOC (personal/social), PHYS (physical), GPE-AFF (GPE affiliation), OTHER-AFF (PER/ORG affiliation), ART (agent-artifact) and DISC (discourse).

Traditionally, the three sub-tasks of end-to-end relation extraction are carried out serially in a “pipeline” fashion. In this case, the errors in any sub-task affect subsequent sub-tasks. Another disadvantage of this “pipeline” approach is that it allows only one-way *information flow*, i.e. the knowledge about entities is used for identifying relations but not vice versa. Hence to overcome this problem, several approaches (Roth and Yih, 2004; Roth and Yih, 2002; Singh et al., 2013; Li and Ji, 2014) were proposed which carried out these sub-tasks jointly rather than in “pipeline” manner.

We propose a new approach which combines

¹www ldc.upenn.edu/sites/www ldc.upenn.edu/files/english-edt-v4.2.6.pdf

²www ldc.upenn.edu/sites/www ldc.upenn.edu/files/english-rdc-v4.3.2.PDF

Entity Mention	Boundaries	Entity Type
His	(0, 0)	PER
sister	(1, 1)	PER
Mary Jones	(2, 3)	PER
United Kingdom	(7, 8)	GPE

Table 1: Expected output of end-to-end relation extraction system for entity mentions

the powers of Neural Networks and Markov Logic Networks to jointly address all the three sub-tasks of end-to-end relation extraction. We design the “All Word Pairs” neural network model (AWP-NN) which reduces solution of these three sub-tasks to predicting an appropriate label for each word pair in a given sentence. End-to-end relation extraction output can then be constructed easily from these labels of word pairs. Moreover, as a separate prediction is made for each word pair, there may be some inconsistencies among the labels. We address this problem by refining the predictions of AWP-NN by using inference in Markov Logic Networks so that some of the inconsistencies in word pair labels can be removed at the sentence level.

The specific contributions of this work are : i) modelling boundary detection problem by introducing a special relation type WEM and ii) a single, joint neural network model for all three sub-tasks of end-to-end relation extraction. The paper is organized as follows. Section 2 provides a detailed problem definition. Section 3 describes our AWP-NN model in detail, followed by Section 4 which describes how the predictions of AWP-NN model are revised using inference in MLNs. Section 5 provides experimental results and analysis. Finally, we conclude in Section 6 with a short note on future work.

2 Problem Definition

Given a sentence as an input, an end-to-end relation extraction system should produce a list of entity mentions within it. For each entity mention, its boundaries and entity type should be identified. Also, for each pair of valid entity mentions, it should decide whether any pre-defined semantic relation holds between them.

Consider the sentence : His₀ sister₁ Mary₂ Jones₃ went₄ to₅ the₆ United₇ Kingdom₈ .₉ Here, end-to-end relation extraction should produce the output as shown in the tables 1 and 2.

Entity Mention Pair	Relation Type
His, sister	PER-SOC
His, Mary Jones	PER-SOC
sister, United Kingdom	PHYS
Mary Jones, United Kingdom	PHYS

Table 2: Expected output of end-to-end relation extraction system for relations

3 All Word Pairs Model (AWP-NN)

We propose a single, joint model for addressing all three sub-tasks of end-to-end relation extraction : i) identifying boundaries of entity mentions, ii) identifying entity types of these mentions and iii) identifying appropriate semantic relation for each pair of mentions. We refer to this model as AWP-NN, i.e. All Word Pairs model using Neural Networks. Here, annotations of all these three sub-tasks can be represented by assigning an appropriate label to each pair of words. It is not necessary to assign label to all possible word pairs; rather i^{th} word is paired with j^{th} word only when $j \geq i$. AWP-NN model is motivated from the table representation idea proposed by Miwa and Sasaki (2014) but differs significantly from it in following ways:

1. boundary identification is modelled with the help of a special relation type (WEM) instead of BIO (**B**egin, **I**nside, **O**ther) encoding or BILOU (**B**egin, **I**nside, **L**ast, **U**nit, **O**ther) encoding
2. neural network model for prediction of appropriate label for each word pair instead of structured prediction

Labels predicted by the AWP-NN model for each word pair can then be used to construct the end-to-end relation extraction output as described in tables 1 and 2.

Consider the example sentence from Section 2. Table 3 shows true annotations of all word pairs in this sentence as required for training the AWP-NN model. Labels used for these annotations can be grouped into the following 5 logical clusters:

1. PER, ORG, GPE, LOC, FAC, VEH and WEA : Represent entity type of *head word* of an entity mention when both the words in a word pair are the same
2. OTH : Represents words which are not head words of any entity mention and both the words in a word pair are the same

	His	sister	Mary	Jones	went	to	the	United	Kingdom	.
His	PER	PER-SOC	NULL	PER-SOC	NULL	NULL	NULL	NULL	NULL	NULL
sister		PER	NULL	NULL	NULL	NULL	NULL	NULL	PHYS	NULL
Mary			OTH	WEM	NULL	NULL	NULL	NULL	NULL	NULL
Jones				PER	NULL	NULL	NULL	NULL	PHYS	NULL
went					OTH	NULL	NULL	NULL	NULL	NULL
to						OTH	NULL	NULL	NULL	NULL
the							OTH	NULL	NULL	NULL
United								OTH	WEM	NULL
Kingdom									GPE	NULL
.										OTH

Table 3: Annotation of all word pairs as per the AWP-NN model

3. EMP-ORG, PHYS, OTHER-AFF, EMP-ORG-R, PHYS-R, OTHER-AFF-R³, PER-SOC, GPE-AFF and ART : Represent relation type between *head words* of any two entity mentions
4. NULL : Indicates that no pre-defined semantic relation exists between the words in the word pair
5. WEM (Within Entity Mention) : Indicates that the words in the word pair belong to the same entity mention and one of the word is the *head word* of that mention

3.1 Features for the AWP-NN model

Previous work (Zhou et al., 2005; Jiang and Zhai, 2007; Bunescu and Mooney, 2005; Qian et al., 2008) in relation extraction establishes the importance of both lexical and syntactic features. Hence, we designed features to capture information about word sequences, POS tags and dependency structure. As each word pair constitutes a separate instance for classification, features are of three types: i) features characterizing individual word in a word pair, ii) features characterizing properties of both the words at a time and iii) features based on feedback, i.e. predictions of preceding instances.

3.1.1 Individual word features

These features are generated separately for both the words in a word pair.

1. Word itself and its POS tag
2. Previous word and previous POS tag
3. Next word and next POS tag
4. Parent / Governor of the word in the dependency tree, the corresponding dependency relation type and POS tag of the parent

³EMP-ORG-R, PHYS-R and OTHER-AFF-R correspond to relation types EMP-ORG, PHYS and OTHER-AFF in the reverse direction, respectively.

3.1.2 Word pair features

These features are generated for a word pair (say $\langle W_i, W_j \rangle$) as a whole.

1. Words distance (WD): Number of words in the sentence between the words W_i and W_j
2. Tree distance (TD): Number of words on the path leading from W_i to W_j in the sentence's dependency tree
3. Common Ancestor (CA): Lowest common ancestor of the two words in the dependency tree
4. Ancestor Position (AP): It indicates the position of the common ancestor with respect to the two words of a word pair. Different possible positions of the ancestor are - left of W_i , W_i itself, between W_i and W_j , W_j itself and right of W_j .
5. Dependency Path (DP_1, DP_2, \dots, DP_K): Sequence of dependency relation types (ignoring directions) on the dependency path leading from W_i to W_j in the sentence's dependency tree.

3.1.3 Feedback features

These features are based on predictions of the preceding instances. Unlike other sequence labelling problems such as Named Entity Recognition where each word gets a label and there is natural order / sequence of instances (i.e. words), there is no natural order / sequence of instances (i.e. word pairs) for AWP-NN model. Hence, for each instance we identify its two preceding instances and define two corresponding feedback features (FB_1 and FB_2). Let $\langle W_i, W_j \rangle$ be an instance representing a word pair in a sentence having N words such that $1 \leq i, j \leq N$ and $i \leq j$. There are following two cases for identifying two preceding instances of $\langle W_i, W_j \rangle$:

- If $i = j$ then both the preceding instances are same i.e. $\langle W_{i-1}, W_{i-1} \rangle$. Feedback features: $FB_1 = FB_2 = \text{LabelOf}(\langle W_{i-1}, W_{i-1} \rangle)$
- If $i < j$ then the preceding instances are

$\langle W_i, W_i \rangle$ and $\langle W_j, W_j \rangle$. Feedback features: $FB_1 = \text{LabelOf}(\langle W_i, W_i \rangle)$ and $FB_2 = \text{LabelOf}(\langle W_j, W_j \rangle)$

Label predictions of the preceding instances are then represented using one-hot encoding and used as features. During training, true labels of the preceding instances are used but while decoding, the predicted labels of these instances are used. Hence during decoding, predictions for word pairs of the form $\langle W_i, W_i \rangle$ (diagonal word pairs in the table 3) are obtained first, starting from $i = 1$ to N . Predictions of other word pairs can be obtained later, as predictions of their preceding instances would then be available.

3.2 Architecture of the AWP-NN model

Figure 1 shows various major components in the architecture of the AWP-NN model.

3.2.1 Embedding Layers

Most of the features used by the model are discrete in nature such as words, POS tags, dependency relation types and ancestor position. These discrete features have to be mapped to some numerical representation and *embedding* layers are used for this purpose. We have employed following embedding layers to represent various types of features:

Word embedding layer: It maps each word to a real-valued vector of some fixed dimensions. We initialize this layer with the *pre-trained* 100 dimensional GloVe word vectors⁴ learned on Wikipedia corpus. All the different features which are expressed in the form of words ($W_1, W_2, NW_1, PW_1, NW_2, PW_2, Pa_1, Pa_2$ and CA in the figure 1) share the *same* word embedding layer. During training, the initial embeddings get *fine-tuned* for our supervised classification task.

POS embedding layer: It maps each distinct POS tag to some real-valued vector representation. All the different features which are expressed in the form of POS tags ($T_1, T_2, NT_1, PT_1, NT_2, PT_2, PaT_1$ and PaT_2 in the figure 1) share the *same* embedding layer.

Dependency relation type embedding layer: It maps each distinct dependency relation type to some real-valued vector representation. Both the features based on dependency types ($DR_1, DR_2, DP_1, \dots, DP_K$ in the figure 1) also share the *same* embedding layer.

⁴<http://nlp.stanford.edu/projects/glove/>

AP embedding layer: It maps each distinct ancestor position to some real-valued vector representation.

WD/TD embedding layer: Even though word distance (WD) and tree distance (TD) are numerical features, we used embeddings to represent each distinct value for them as range of values of these features is large. It was observed to be better than directly providing them as inputs to the neural network.

In our experiments, we used 20 dimensions for POS embeddings, 40 for dependency relation type embeddings and 5 dimensions for AP, WD and TD embeddings. Unlike word embeddings these were initialized randomly during training.

3.2.2 Hidden Layers

First hidden layer is divided in 3 parts. First two parts of 60 nodes each are connected to only the features capturing first and second word, respectively. These nodes are expected to capture higher level abstract features of both the words separately. In order to force these two parts to learn similar abstract features, the weights matrix is shared among them. The third part of the first hidden layer consisting of 500 nodes is connected to all the input features except dependency path, i.e. individual word features of two words, word pair features and feedback features. Output of this part is further given as input to the second hidden layer of 250 units. Output of the second hidden layer is fed to the final softmax layer. Also, outputs of the first two parts of the first hidden layer are directly connected to the final softmax layer. As the dependency path is represented as a sequence of dependency relation types, it is fed to a separate LSTM layer. Output of the LSTM layer is directly connected to the final softmax layer. Softmax layer consists of 19 nodes, each representing one of the possible prediction label described earlier.

4 Inference using Markov Logic Networks

Pawar et al. (2016) presented an approach for end-to-end relation extraction which uses Markov Logic Networks (MLN) (Richardson and Domingos, 2006) to obtain *globally consistent* output by combining *local* outputs of individual classifiers. They developed separate classifiers for identifying mention boundaries, predicting entity types and

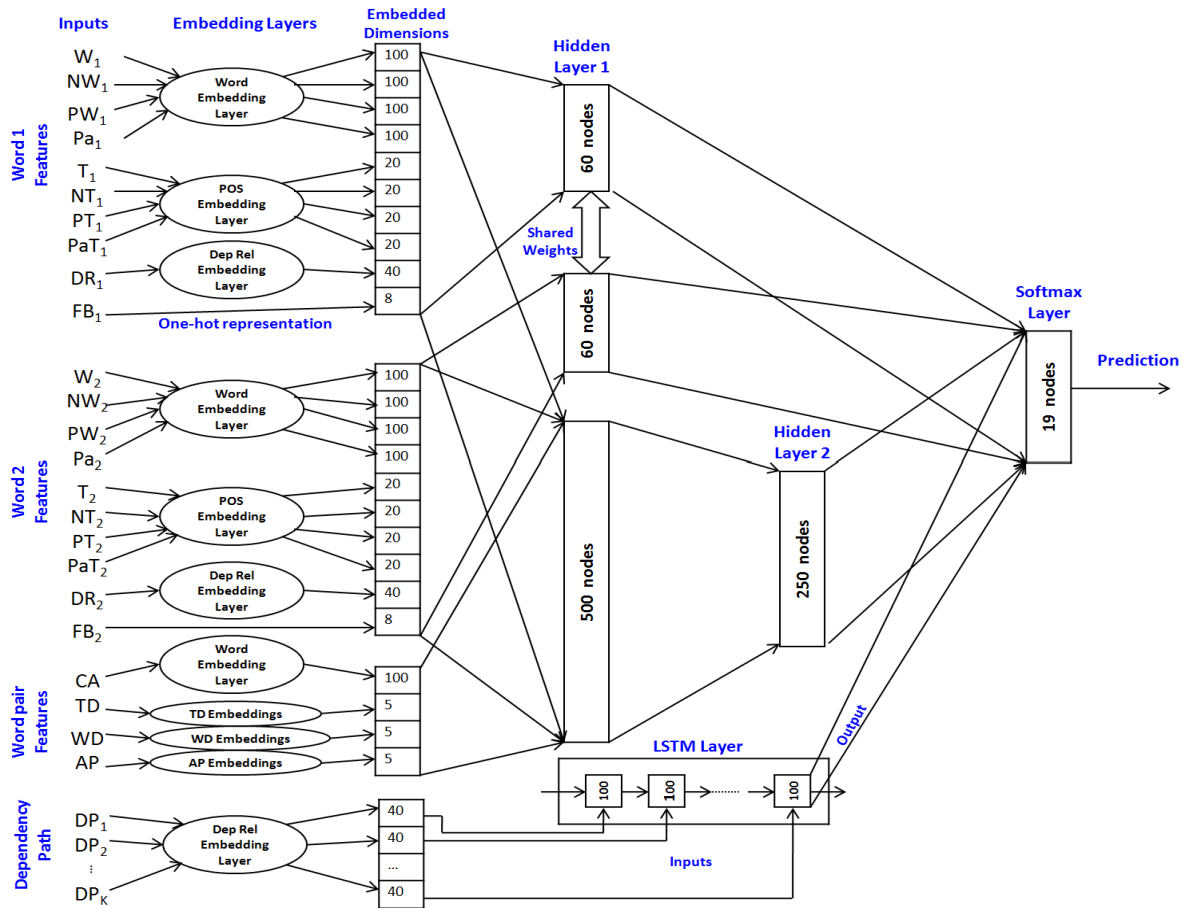


Figure 1: AWP-NN model architecture for predicting appropriate label for the given word pair. (W_1, W_2 : words in the word pair; $NW_1, PW_1, NW_2, PW_2, NT_1, PT_1, NT_2, PT_2$: next and previous words/POS tags of W_1 and W_2 ; Pa_1, DR_1, Pa_2, DR_2 : parents and corresponding dependency relation types of W_1 and W_2 in the dependency tree; PaT_1, PaT_2 : POS tags of the parents of W_1 and W_2 ; FB_1, FB_2 : Predictions of the preceding instances; CA : Lowest common ancestor of W_1 and W_2 in the dependency tree; TD : Tree distance; WD : Words distance; AP : Ancestor position; DP_1, DP_2, \dots, DP_K : Sequence of dependency relation types on the dependency path leading from W_1 to W_2 ; Embedding layers for words, POS and dependency relations are shown separately for clarity, but are shared throughout the network.

predicting relation types. Outputs of these classifiers may be inconsistent. E.g., if PER-SOC relation is predicted by the local relation classifier for an entity pair and the local entity classifier predicts entity type as ORG for one of the entity mentions, then there is an inconsistency. Because PER-SOC relation can only exist between two PER entity mentions. Such domain knowledge can be easily incorporated in the form of first-order logic rules in MLN. For each sentence, predictions of individual classifiers are represented in an MLN as first-order logic rules where weights of these rules are proportional to the prediction probabilities. The consistency constraints among the relation types and entity types can be represented in the form of first-order logic rules with infinite weights. Now,

the inference in such an MLN generates a globally consistent output with maximum weighted satisfiability of the rules.

AWP-NN is a single joint model which captures boundaries of mentions, their types and relations among them. As the same parameters are shared for all entity as well as relation type predictions, we expect the model to learn dependencies among relation and entity types. However, as it makes separate predictions for each word pair, there might be some inconsistencies among the labels as described above. We adopt the MLN-based approach of Pawar et al. (2016) for handling these inconsistencies and generate a globally consistent output. For this adoption, we consider the AWP-NN predictions for the words pairs where a word

<p>Domains: Let N be the number of words in the sentence in consideration.</p> $word = \{1, 2, \dots, N\}, etype = \{PER, ORG, LOC, GPE, WEA, FAC, VEH, OTH\}$ $rtype = \{EMPORG, GPEAFF, OTHERAFF, PERSOC, PHYS, ART, NULL, WEM\}$
<p>Evidence Predicates: $ET(word, etype) : \text{AWP-NN predictions for word pairs } \langle W_i, W_j \rangle \text{ where } i = j$ $RT(word, word, rtype) : \text{AWP-NN predictions for word pairs } \langle W_i, W_j \rangle \text{ where } i < j$</p>
<p>Query Predicates: $ETFinal(word, etype) : \text{Global predictions for word pairs } \langle W_i, W_j \rangle \text{ where } i = j$ $RTFinal(word, word, rtype) : \text{Global predictions for word pairs } \langle W_i, W_j \rangle \text{ where } i < j$</p>
<p>Some examples of generic rules:</p> $RTFinal(x, y, EMPORG) \wedge ETFinal(x, PER) \Rightarrow (ETFinal(y, ORG) \vee ETFinal(y, GPE)).$ $RTFinal(x, y, PERSOC) \Rightarrow (ETFinal(x, PER) \wedge ETFinal(y, PER)).$

Table 4: Domains and Predicates used for constructing MLN for any given sentence

is paired with itself (diagonal entries in table 3), as *entity* type predictions. Whereas all other word pairs where a word is paired with any subsequent word in the sentence, are considered as *relation* type predictions. Table 4 describes the domains and predicates required for generating an MLN for any given sentence. Unlike Pawar et al. (2016) which considers all predicted mentions in their *entity* domain, we consider all words in our *word* domain. But to keep the size of the MLN in check, we keep only those words in the *word* domain which are part of *interesting* word pairs. A word pair is an *interesting* word pair, if it can potentially represent a relation i.e. if AWP-NN model assigns a probability more than some threshold (say 0.01) for any non-NULL relation type. All the *generic rules* (with infinite weights) described in (Pawar et al., 2016) are used for imposing constraints among the relation and entity types. Also, we added following additional *generic rules* for specifying constraints for our *WEM* relation type, which captures information about mention boundaries.

$$RTFinal(x, y, WEM) \Rightarrow (ETFinal(x, OTH) \vee ETFinal(y, OTH)).$$

$$RTFinal(x, y, WEM) \Rightarrow (!ETFinal(x, OTH) \vee !ETFinal(y, OTH)).$$

By definition, the WEM relation holds between a head word of an entity mention and other words of that entity mention. Additionally, head word of an entity mention is labelled with appropriate entity

label and other words are labelled with entity type OTH. The above rules state that if there is WEM relation between two words x and y then at least one of them should have label OTH and at least one of them should have entity type label, i.e. a label from domain *etype* other than OTH.

Similarly, all the *sentence-specific rules* (with finite weights proportional to AWP-NN prediction probabilities) described in (Pawar et al., 2016) are also generated for representing predictions of the AWP-NN model. We use *Constant Multiplier* (CM) as the weights assignment strategy. Following rule would be generated for each entity type E (from *etypes*) for each word pair $\langle W_i, W_j \rangle$, with the weight $10 \cdot Pr_{AWP-NN}(E|\langle W_i, W_j \rangle)$ where E_{max} is the predicted entity type with the highest probability:

$$ET(i, E_{max}) \Leftrightarrow ETFinal(i, E)$$

Similarly, following rule would be generated for each relation type R (from *rtypes*) for each word pair $\langle W_i, W_j \rangle$, with the weight $10 \cdot Pr_{AWP-NN}(R|\langle W_i, W_j \rangle)$ where R_{max} is the predicted relation type with the highest probability:

$$RT(i, R_{max}) \Leftrightarrow RTFinal(i, R)$$

Using these *generic* and *sentence-specific* rules, an MLN is constructed for each sentence. The best values of *ETFinal* and *RTFinal* (query predicates) for each word pair are obtained by using the

inference in this MLN with *ET* and *RT* as evidence predicates based on AWP-NN’s predictions.

5 Experimental Analysis

5.1 Dataset

ACE 2004 dataset (Dodding et al., 2004) is the most widely used dataset⁵ for reporting relation extraction performance. We use this dataset to demonstrate effectiveness of our approach for end-to-end relation extraction using AWP-NN model and MLN inference. We perform 5-fold cross-validation on this dataset where the folds are formed at the document level. We follow the same assumptions made by (Chan and Roth, 2011; Li and Ji, 2014; Pawar et al., 2016), which are - ignore the DISC relation, do not consider implicit relations (resulting due to intra-sentence co-references) as false positives and use coarse-level entity and relation types.

Direction of Relations: Out of 6 coarse-level relation types that we are considering, we need not model direction for relation types PER-SOC, GPE-AFF and ART. Because in case of these relations, given the entity types of their arguments, the direction of relation is not necessary or becomes implicit. As PER-SOC is a social relation between two PER entity mentions, the direction is not necessary. For GPE-AFF, as entity type of one of the arguments is always GPE, the direction becomes implicit. Also, the relation type ART always holds between an agent (PER, ORG or GPE) and an artifact (FAC, WEA or VEH), hence the direction is implicit. Whereas for relation like EMP-ORG which also represents subsidiary relationship between two ORG entity mentions, it is important to model the relation direction explicitly. Consider following sentence fragments:

- ..the fisheries section of the Gulf Coast Research Laboratory..
- ..company that owned Road & Track..

Here, EMP-ORG relation exists between ORG entity mentions `fisheries section` and `Gulf Coast Research Laboratory`. Whereas, EMP-ORG-R relation holds between `that` and `Road & Track`.

Hence, we consider 9 distinct relation types: EMP-ORG, EMP-ORG-R, PHYS, PHYS-R, OTHER-AFF, OTHER-AFF-R, PER-SOC,

⁵We haven’t yet acquired a more recent ACE 2005 dataset

GPE-AFF and ART. Hence, the overall dataset contained 4074 instances⁶ of valid relation types.

5.2 Implementation details

We used Keras (Chollet, 2015) for implementing our AWP-NN model. The model was trained for 40 epochs using batch size of 64 instances. We used *Dropout* (Srivastava et al., 2014) for regularization with probability 0.5 for hidden layers and 0.1 for embedding layers. We used the tool *Alchemy*⁷ for MLN inference. The value of K (maximum length of dependency path, see Figure 1) was set to be 4, hence all word pairs having length of dependency path more than 4 were assumed to have NULL label.

5.3 Results

Table 5 shows the comparative performances (in terms of micro-F1 measure) for various approaches. The results are divided in three different sections:

1. **only entity extraction:** It includes boundary identification as well as entity type classification.
2. **only relation extraction:** It includes relation type classification for each pair of predicted entity mentions. It is a relaxed version of end-to-end relation extraction problem where correct relation label for an entity mention pair is counted as a true positive even if entity types of one or both the mentions are identified incorrectly.
3. **entity+relation extraction:** It is end-to-end relation extraction which includes boundary identification, entity type classification and relation type classification. Here, correct relation label for an entity mention pair is counted as a true positive only if boundaries and entity types of both the mentions are identified correctly.

It can be observed in the table 5 that end-to-end relation extraction performance of our AWP-NN model is better than all the 4 previous approaches (Chan and Roth, 2011; Li and Ji, 2014; Pawar et al., 2016; Miwa and Bansal, 2016) on the ACE 2004 dataset. However, the AWP-NN+MLN approach which uses MLN inference to revise AWP-NN predictions during decoding, achieves the best performance.

⁶279 instances of type DISC were not considered. Additionally, 21 relation instances were not contained in a single sentence as per our sentence detection algorithm.

⁷<https://alchemy.cs.washington.edu/>

Approach	Entity Extraction			Relation Extraction			Entity+Relation		
	P	R	F	P	R	F	P	R	F
(Chan and Roth, 2011)				42.9	38.9	40.8			
(Li and Ji, 2014)	83.5	76.2	79.7	64.7	38.5	48.3	60.8	36.1	45.3
(Pawar et al., 2016)	79.0	80.1	79.5	57.9	45.6	51.0	52.4	41.3	46.2
(Miwa and Bansal, 2016)	80.8	82.9	81.8				48.7	48.1	48.4
AWP-NN	81.1	79.7	80.4	60.3	48.1	53.5	55.6	44.4	49.3
AWP-NN + MLN	81.2	79.7	80.5	61.1	47.9	53.7	56.7	44.5	49.9

Table 5: Performance of various approaches on the ACE 2004 dataset. The numbers are micro-averaged and obtained after 5-fold cross-validation. Actual folds used by each algorithm may differ.

5.3.1 Statistical Significance

As neural networks are initialized randomly, if we train a neural network model multiple times, different predictions are obtained each time. Hence, it is important to establish the statistical significance of the performance. We train our AWP-NN model 30 times independently and obtain 30 different values for precision, recall and F1 score. The numbers shown in table 5 are average values over these 30 runs. Also, in order to establish that the F1 score of AWP-NN model is significantly higher than the best previous F1 score of 48.4% (by Miwa and Bansal (2016)), we conduct one tailed one sample t-test. Here, mean and standard deviation of sample of 30 F1 scores by AWP-NN are 49.3 and 0.44, respectively. This leads to p-value of 1.23×10^{-12} , hence establishing the statistical significance of AWP-NN’s performance.

5.4 Analysis of results

5.4.1 Effect of using MLN

We analyzed the effect of using MLN by observing the individual sentences where errors of AWP-NN were being corrected by MLN. As an example, consider the following sentence:

Lemieux₀ rescued₁ his₂ team₃ from₄ bankruptcy₅ last₆ season₇ by₈ exchanging₉ deferred₁₀ salary₁₁ for₁₂ an₁₃ ownership₁₄ stake₁₅ .₁₆

End-to-end relation extraction output produced by the AWP-NN model for this sentence is shown in the tables 6 and 7. Only error in this output is that entity type of the mention `team` should be `ORG` instead of `PER` as it refers to some professional team. After MLN inference, the entity type of `team` is corrected to `ORG`. This happens because of high-confidence `EMP-ORG` relations between `Lemieux` and `team` and between `his` and `team`. As both `Lemieux` and `his` are of type

`PER` with high confidence, global inference using `MLN`⁸ forces type of `team` to be `ORG` to ensure compatibility of relation and entity types.

Entity Mention	Boundaries	Entity Type
Lemieux	(0, 0)	PER
his	(2, 2)	PER
team	(3, 3)	PER

Table 6: End-to-end relation extraction output (entity mentions) produced by the AWP-NN model

Entity Mention Pair	Relation Type
Lemieux, team	EMP-ORG
his, team	EMP-ORG

Table 7: End-to-end relation extraction output (relations) produced by the AWP-NN model

The AWP-NN model was able to outperform (see table 5) all 4 previous approaches without the help of MLN. One reason behind this may be that the AWP-NN model itself was sufficient to learn most of the dependencies among the entity and relation types. However, MLN helped to improve the performance of AWP-NN by 0.6 F1. Though considerable improvement was observed in the precision value, the recall improvement was not significant. In other words, MLN was observed to be more effective for reducing false positives than false negatives.

5.4.2 Difficult to identify entities

We observed that for some entity mentions, it is very difficult to identify their entity types as the key information required for identification lies outside the current sentence. Currently, our approach does not use any information outside the sentences, such as document level co-reference

⁸Detailed MLN rules & inference results for this sentence can be found at: www.cse.iitb.ac.in/~sachinpawar/MLN/sentence.html

information. Usually these difficult to classify entity mentions are pronoun mentions. Some examples are as follows:

1. Though, I think that if they could stifle the entire peace process at the moment, then that is what they'd like to do.
2. It is a partially victory for both sides.

Here, in the first sentence, it is difficult to identify (even for humans) whether entity type of they is PER (e.g. set of leaders) or GPE (e.g. countries). Also, in the second sentence, entity type of sides can be any of PER, ORG or GPE depending on the context. In future, we plan to capture document level information for correctly predicting types of such mentions.

6 Related Work

There have been multiple lines of research for jointly modelling and extracting entities and relations. Integer Linear Programming (ILP) based approaches (Roth and Yih, 2004; Roth and Yih, 2007) were the earliest ones. Here, various local decisions are associated with suitable “cost” values and they are represented using an integer linear program. The optimal solution to this integer linear program provides the best global output. Another significant lines of research were Probabilistic Graphical Models (Roth and Yih, 2002; Singh et al., 2013), Card-pyramid parsing (Kate and Mooney, 2010) and Structured Prediction (Li and Ji, 2014; Li et al., 2014; Miwa and Sasaki, 2014).

Four previous approaches (Miwa and Sasaki, 2014; Li and Ji, 2014; Pawar et al., 2016; Miwa and Bansal, 2016) are the most similar to our approach in the sense that they all address the problem of end-to-end relation extraction without assuming gold-standard entity mention boundaries like the earlier approaches. Our idea of labelling “all word pairs” is similar to the table representation idea of Miwa and Sasaki (2014). The major difference is that they identify boundaries of mentions through BIO encoding of labels whereas we try to capture boundaries by treating them as an additional relation type WEM. Also, they perform structured prediction with beam search to find optimal label assignment to the table, whereas we opt for neural network based classification. The idea of using MLNs to incorporate domain knowl-

edge and perform joint inference to obtain globally consistent output was proposed by Pawar et al. (2016). The current state-of-the-art approach for end-to-end relation extraction is by Miwa and Bansal (2016), who employ LSTM-RNN based model for addressing this problem.

7 Conclusion and Future Work

We proposed a novel approach for end-to-end relation extraction which carries out its all three sub-tasks (identifying entity mention boundaries, their entity types and relations among them) jointly by using a neural network based model. We proposed a “All Word Pairs” neural network model (AWP-NN) which reduces solution of these three sub-tasks to predicting an appropriate label for each word pair in a given sentence. End-to-end relation extraction output is then constructed from these labels of word pairs. We further improved output of the AWP-NN model by using inference in Markov Logic Networks so that some of the inconsistencies in word pair labels can be removed at the sentence level.

We demonstrated effectiveness of our approaches (AWP-NN and AWP-NN+MLN) on the standard dataset of ACE 2004. They outperformed all 4 previously reported joint modelling approaches (Chan and Roth, 2011; Li and Ji, 2014; Pawar et al., 2016; Miwa and Bansal, 2016) for end-to-end relation extraction. Since all three sub-tasks share the same AWP-NN model parameters, many inter-task dependencies are captured effectively by the AWP-NN itself (without MLN) and this can be validated by the fact that AWP-NN itself performs better than all other joint models. However, MLN certainly helps to further improve the end-to-end relation extraction performance by correcting some errors in predictions of the AWP-NN model.

In future, we plan to incorporate some additional features (e.g. document level co-reference information) in the AWP-NN model for improving its performance further. Also, deeper analysis of the errors is required to have a better understanding about which characteristics are better captured by the AWP-NN model as compared to the MLN and vice versa. This will help these two to complement each other in a better way.

References

- Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.
- Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Franois Chollet. 2015. keras. <https://github.com/fchollet/keras>.
- George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. In *LREC*, volume 2, page 1.
- Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120, Rochester, New York, April. Association for Computational Linguistics.
- Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden, July. Association for Computational Linguistics.
- Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. Association for Computational Linguistics.
- Qi Li, Heng Ji, Yu HONG, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1846–1851, Doha, Qatar, October. Association for Computational Linguistics.
- Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany, August. Association for Computational Linguistics.
- Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, October. Association for Computational Linguistics.
- Sachin Pawar, Pushpak Bhattacharyya, and Girish Palshikar. 2016. End-to-end relation extraction using markov logic networks. In *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*, LNCS 9624. Springer.
- Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 697–704, Manchester, UK, August. Coling 2008 Organizing Committee.
- Matthew Richardson and Pedro Domingos. 2006. Markov Logic Networks. *Machine learning*, 62(1-2):107–136.
- Dan Roth and Wen-tau Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. ACL.
- Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8, Boston, Massachusetts, USA, May 6 - May 7. Association for Computational Linguistics.
- Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to statistical relational learning*, pages 553–580.
- Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6. ACM.
- Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.
- GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Trust, but Verify!

Better Entity Linking through Automatic Verification

Benjamin Heinzerling*
AIPHES
Heidelberg Institute for
Theoretical Studies

benjamin.heinzerling@h-its.org

Michael Strube
Heidelberg Institute for
Theoretical Studies
michael.strube@h-its.org

Chin-Yew Lin
Microsoft Research
cyl@microsoft.com

Abstract

We introduce automatic verification as a post-processing step for entity linking (EL). The proposed method *trusts* EL system results collectively, by assuming entity mentions are mostly linked correctly, in order to create a semantic profile of the given text using geospatial and temporal information, as well as fine-grained entity types. This profile is then used to automatically *verify* each linked mention individually, i.e., to predict whether it has been linked correctly or not. Verification allows leveraging a rich set of global and pairwise features that would be prohibitively expensive for EL systems employing global inference. Evaluation shows consistent improvements across datasets and systems. In particular, when applied to state-of-the-art systems, our method yields an absolute improvement in linking performance of up to 1.7 *F1* on AIDA/CoNLL'03 and up to 2.4 *F1* on the English TAC KBP 2015 TEDL dataset.

1 Introduction

Entity linking (EL) is the task of automatically linking mentions of entities such as persons, locations, or organizations to their corresponding entry in a knowledge base (KB). The task is generally approached by generating a set of candidate entities¹ for a given mention and then ranking those candidates. Approaches differ in whether they rank a mention's candidates independently of the candidates of other mentions ("local inference") or

whether they rank all candidates of all mentions simultaneously by incorporating a global coherence measure into the optimization goal ("global inference").

While linguistically well-founded in the concept of lexical cohesion (Halliday and Hasan, 1976), global inference approaches (Kulkarni et al., 2009; Hoffart et al., 2011a) do not scale well with number of mentions and number of candidate entities. In contrast, local approaches do not suffer from scalability issues, since they only optimize the similarity between mention context and candidate KB entry text (Bunescu and Paşca, 2006; Cucerzan, 2007), usually also including a popularity prior² (Milne and Witten, 2008; Spitkovsky and Chang, 2012). Recent local approaches achieve state-of-the-art results by using convolutional neural networks to capture similarity at multiple context sizes (Francis-Landau et al., 2016), but, by definition, fail to take global coherence into account.

To avoid the trade-off between the efficiency of local inference on the one hand and the coherence benefits of global inference on the other, we propose a two-stage approach: In the first stage, candidate entities are ranked by a fast, local inference-based EL system. In the second stage these results are used to create a semantic profile of the given text, derived from rich data the KB contains about the top-ranked candidates. Since the linking precision of current EL systems is relatively high, we trust that this profile is reasonably accurate and leverage it to measure the cohesive strength between a given candidate entity and the other linked entities mentioned in the text. We then automatically verify the first stage results by classifying entity links as correct if they display high coherence, and as wrong if there are only weak or no cohesive

*The majority of this work was done during an internship at Microsoft Research Asia.

¹We use *entity* to refer to both real-word entities and to their corresponding entries in the KB.

²Also referred to as *commonness prior* by some authors.

ties to the semantic profile. Verification results can be used in at least three ways:

1. To increase linking precision by filtering out all entity links classified as wrong;
2. To rerank candidate entities by the class probability estimated by the verifier, i.e., prefer candidates that were predicted as correct with higher probability; or
3. To employ a more sophisticated EL system to re-link all entity links classified as wrong, using the entity links deemed correct as additional context.

In this work we investigate options 1. and 2., and make the following contributions:

- We propose automatic verification as a post-processing step for EL systems;
- We propose global coherence features based on notions of entity type coherence, geographic coherence, and temporal coherence;
- We show how these novel features, as well as features developed in prior work, can be used to verify EL results; and
- We show that automatic verification consistently improves linking performance in an evaluation across two datasets and seven different EL systems.

2 Method

We cast entity linking verification as a supervised classification task. Given EL system output on a training set with gold standard linked entity annotations, we extract global, pairwise, and local features and train a classifier to predict whether a given mention has been linked correctly by the EL system.

In the standard EL setting, global inference is an NP-hard problem, since all combinations of all candidate entities of all mentions are considered simultaneously. In our proposed automatic verification setting, however, taking only the top candidate entities into account allows us to employ knowledge-rich, global coherence features that would be prohibitively expensive otherwise.

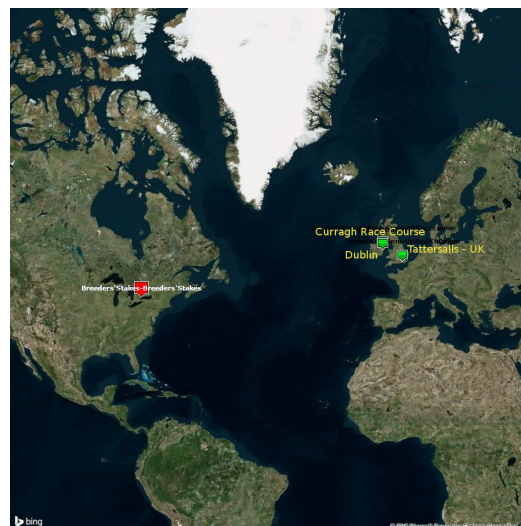


Figure 1: Example showing a geographical outlier: Breeder’ Stakes (red, in Canada) and contextual entities located in Ireland and the UK (green).

2.1 Aspects of Global Coherence

Global coherence captures how well a candidate entity fits into the overall semantic profile of a text. Current global inference approaches optimize a single coherence measure, most commonly a measure of general semantic relatedness such as the Milne-Witten distance (Milne and Witten, 2008), or keyphrase overlap relatedness (KORE) (Hoffart et al., 2012).

In contrast, verification allows employing many global coherence features, which we categorize according to four aspects of coherence: geographical coherence and temporal coherence, which to our knowledge have not been used before in EL, as well as entity type coherence and the general semantic relatedness mentioned above.

2.1.1 Geographic Coherence

Entities mentioned in a text tend to be geographically close or clustered around very few locations. We use this observation to identify geographic outliers as potential entity linking mistakes.

For example, consider the mention *Breeders Stakes* in the following excerpt (CoNLL 1112testa):

DUBLIN 1996-08-31 Result of the Tattersalls Breeders Stakes , a race for two-year-olds run over six furlongs at The Curragh ...

DUBLIN, Tattersalls (a company doing business in the UK and Ireland), and *The Curragh* (a

Predicate
:location.location.geolocation
:organization.organization.geographic_scope
:time.event.locations
:sports.sports_team.location
:organization.organization.headquarters

Table 1: Freebase predicates for querying geo-coordinates of locations, geo-political entities, and organizations.

Predicate
:people.person.date_of_birth
:organization.organization.date_founded
:sports.sports_team.founded
:location.dated_location.date_founded
:time.event.start_date
:film.film.initial_release_date
:music.album.release_date
:music.release.release_date
:architecture.structure.construction_started
:architecture.structure.opened
:people.deceased_person.date_of_death
:location.dated_location.date_dissolved
:time.event.end_date
:business.defunct_company.ceased_operations
:architecture.structure.closed

Table 2: Freebase predicates for querying the *begin* (top) and *end* (bottom) of an entity’s temporal range.

horse race track in Ireland) clearly situate the text in Ireland (cf. Figure 1). However, some current EL systems link *Breeder Stakes* to the Wikipedia article about the Canadian horse race of the same name, since the Irish race does not have a Wikipedia article and other evidence³ suggests a strong match.

We aim to identify these kinds of errors by first querying locations (Table 1) of all linked mentions in the document, and then performing geographic outlier detection⁴. This yields a binary feature indicating whether a candidate entity is a geographic outlier or not.

Since outliers are rare and hence the resulting features sparse, we also add a feature for the average geographic distance $\bar{d}(e, D)$ of a candidate entity e to all other entities in document D :

$$\bar{d}(e, D) = \frac{\sum_{e' \in D \setminus e} d(e, e')}{|D| - 1}$$

where $d(e, e')$ is the geographic distance between entities e and e' , and $|D|$ is the number of entities

³Specifically, high context-similarity due to the appositive *race*, and an almost perfect string match between mention and Wikipedia title.

⁴We use an ensemble of standard outlier detection algorithms provided by the ELKI clustering toolkit (Achter et al., 2011).

mentioned in D . This feature is based on the intuition that a candidate entity which is geographically closer to other entities is more likely to be correct than a distant one.

Geographic scope varies across documents. For example, entities mentioned in a text about world politics will be geographically more distant than entities in a text about a local business). As a scale-invariant distance measure $s(e, D)$, we divide the average distance $\bar{d}(e, D)$ by the average distance between all other entities:

$$s(e, D) = \bar{d}(e, D) / \frac{\sum_{e', e'' \in D \setminus e} d(e', e'')}{|e', e'' \in D \setminus e|}$$

2.1.2 Temporal Coherence

Applying the notion of coherence to the temporal dimension, we observe that entities mentioned in a text tend to be temporally close or clustered around a few points in time.

Entities are associated with temporal ranges with a *begin*, i.e. the point in time at which the entity comes into existence, and an *end*, i.e. the point in time at which the entity ceases to exist. Using the same approach as in geographical outlier detection, we perform temporal outlier detection on all *begin* and *end* times associated with linked entities in the given text, and declare a candidate entity as a temporal outlier if both its *begin* and *end* were detected as outliers.

Since temporal outliers are rare, we also add a feature aiming to capture temporal proximity and distance in a softer fashion with higher coverage; by calculating the total overlap $T(e, D)$ between the temporal range $t(e)$ of a candidate entity e , and the known temporal ranges of all other linked entities in the document D :

$$T(e, D) = \sum_{e' \in D \setminus e} |t(e) \cap t(e')|$$

where $|t(e) \cap t(e')|$ is the length of the overlap between the temporal ranges of entities e and e' .⁵

Analogously to the geographic distance feature, we take temporal proximity, i.e. a large overlap with other temporal ranges, as evidence for a correctly linked entity, and temporal distance, i.e. only small or no overlap with other temporal

⁵We also extract this feature normalized by the number of entity mentions in the document, but did not see any effect. This is likely due to little variation in the number of entities per document for which the KB contains temporal information.

ranges, as evidence for a linking mistake. Temporal ranges are queried from the KB using the predicates shown in Table 2.

The final feature using temporal information checks whether an entity’s temporal ranges contains the document’s creation date. This feature is based on the intuition that, especially in the news genre, an existing entity is more likely to be mentioned than an entity that has already ceased to exist or did not exist at the time of writing. The document creation date is either trivially obtained if metadata is present, or heuristically by using the first date found in the document text by the Heidelberg Time temporal tagger (Strötgen and Gertz, 2010).

2.1.3 Entity Type Coherence

Frequency statistics of the types of entities mentioned in a text are an indicator of what the text is about. For example, looking at the entity type distribution shown in Table 3, we can tell that the corresponding text appears to be about rugby teams. Unlike other methods for representing the “aboutness” of a text, such as topic models, entity type statistics are grounded in the KB, thus offering a simple method of measuring the relatedness between entities in terms of their types via the similarity of their type distributions.

Specifically, we model entity type coherence between a given candidate entity e and all other linked entities in document D as the cosine similarity of the respective type distributions. Type frequencies are TF-IDF weighted, in order to discount frequent types (e.g. `:base:tagit:concept`) and give more importance to salient types occurring in the document (e.g. `:base.rugby.rugby_club`):

$$coh_{type}(e, D) = sim(types(e), tfidf(types(D)))$$

where sim is the cosine similarity, $types(e)$ a binary vector indicating the types of entity e , and $types(D)$ a vector whose entries are occurrence counts of entity types in document D , which are weighted by $tfidf$.

TF-IDF	Count	Type
1115.67	2	<code>:base.rugby.rugby_club</code>
243.62	3	<code>:organization.organization</code>
231.76	2	<code>:base.schemastaging.sports_team_extra</code>
183.49	2	<code>:sports.sports_team</code>
56.34	2	<code>:base.tagit.concept</code>

Table 3: Entity type distribution in a document about rugby, sorted by type TF-IDF.

2.1.4 Semantic Relatedness

Measures of generic semantic relatedness are a standard feature in global inference systems. We add features for the average and maximum semantic relatedness $SemRel(e, D)$ of a candidate entity e with respect to all other entities e' mentioned in document D , using two semantic relatedness measures:

$$SemRel_{max}(e, D) = max_{e' \in D \setminus e} SemDist(e, e')$$

$$SemRel_{avg}(e, D) = avg_{e' \in D \setminus e} SemDist(e, e')$$

where max and avg are the maximum and average operators. $SemDist$ denotes either the Milne-Witten Distance (Milne and Witten, 2008), which defines relatedness of Wikipedia entries in terms of shared incoming article links, or the Normalized Freebase Distance (Godin et al., 2014), an adaptation of the Milne-Witten Distance to Freebase entities.

2.2 Pairwise Features

Semantic relation: Given a pair consisting of a candidate entities and an entity mention in its context, we add a feature encoding whether a (and if yes which) semantic relation exists between the two entities. We add different features depending on the type of context in which the entity pair occurs: in the same sentence, within a fixed token window, and within the same noun phrase. For example, in the noun phrase *German Chancellor Angela Merkel*, we find a `wasBornIn` and a `isLeaderOf` relation between YAGO entities `ANGELA_MERKEL`⁶ and `GERMANY`. We expect this feature to be sparse, but strong evidence for both arguments of the identified relation being linked correctly. We record the relation type, as some relations tend to be more informative than others, e.g., the `playsFor` relation, which holds between players and sports teams,

⁶In this work, `SMALL_CAPS` denote both real-world entities and their corresponding entries in the knowledge base.

should provide stronger evidence than the less specific `isCitizenOf` relation, which holds between citizens and countries.

Person name consistency: Having observed that some local inference systems tend to make the mistake of linking a full name mention (e.g. “John Smith”) to one entity, and a coreferent surname-only mention (“Smith”) to a different one, we add a binary feature that indicates whether a candidate entity assigned to a partial person name mention agrees with its unambiguous full name antecedent.

2.3 Local Features

Since the global and pairwise features do not have high enough coverage to provide evidence for all linked candidate entities, we employ local features that are devised to capture similarity between a candidate entity and its textual context. As these features are commonly used in EL systems, we only give brief descriptions for completeness.

Popularity prior: The prior probability of the candidate entity given its mention, obtained from the CrossWikis dictionary (Spitkovsky and Chang, 2012). This feature aims to cover unambiguous and almost unambiguous mentions.

Entity type agreement: A binary feature indicating whether the candidate entity type, as found in the KB agrees with the named entity type, as determined by the NER system during preprocessing.

Keyphrase match: Knowledge bases contain various sources of key phrases, such as labels and aliases of semantic types, or salient noun phrases in description texts, e.g., noun phrases occurring in the first, defining sentence of a Wikipedia article. We add a binary feature indicating whether a known keyphrase occurs in the context of a given candidate entity.

Demonym match: This binary feature indicates whether a mention is a demonym of its linked entity, e.g., the mention text *French* is a demonym match for the entity FRANCE.

Mention-entity string match: Finally, we extract features from the string similarity between a mention and the known labels and aliases of a candidate entity. The similarity measures include exact match, case-insensitive match, head match, match with stop words filtered, fuzzy string match, Levenshtein distance, and abbreviation pattern matches, as well as different combinations of these.

Dataset	CoNLL	TAC15
Entity Type	99.2	98.5
Geographic	62.9	41.5
Temporal	87.6	79.4

Table 4: KB coverage of our proposed global coherence features. Shown are the percentages of in-KB mentions in each dataset for which the KB (YAGO or Freebase) contains the required information for each coherence feature set.

3 Experiments

We evaluate our automatic verification method by applying it to the entity linking results produced by seven systems on two standard datasets: CoNLL, which consists of 1393 Reuters news articles annotated with Wikipedia links by Hoffart et al. (2011a) and TAC15, which comprises 315 news articles and discussion forum texts annotated with Freebase links for the TAC KBP 2015 TEDL shared task (Ji et al., 2015).

The KB coverage for each of our proposed global coherence features on these two datasets is shown in Table 4. YAGO and Freebase contain entity type information for almost all in-KB entities mentioned in the two datasets. Geographic data is available for 62.9 percent on CoNLL, but only for 41.5 percent of entities mentioned in TAC15. This difference is likely due to the large fraction of documents from the sports genre in CoNLL. These documents include match result tables mentioning a large number of sports teams, which can be easily located via their cities and stadiums. Temporal information is present for most entities.

Our evaluation uses results of the following EL systems:

AIDA (Hoffart et al., 2011a): This system globally optimizes a graph-based model incorporating three factors: a popularity prior, the context similarity of mention and candidate entity, and coherence modeled via general semantic relatedness measures. We use the AIDA system output on the CoNLL dataset as provided by the Wikilinks project.⁷

SPOTL (Daiber et al., 2013): DBpedia Spotlight is a local inference system. We use results obtained from the Spotlight webservice.⁸

⁷https://github.com/wikilinks/conll103_nel_eval

⁸<https://github.com/dbpedia-spotlight/>

FL (Francis-Landau et al., 2016): This local inference system models mention and entity context with a convolutional neural network (CNN). The CNN captures semantic similarity of a given mention’s context at different granularities (small context window, paragraph, document) and the entity context derived from the entity’s Wikipedia page.

PH (Perschina et al., 2015): This global inference system applies Personal PageRank to a graph whose nodes represent candidate entities and whose edges indicate if a link between the corresponding Wikipedia articles exists. PH achieves the best CoNLL performance among the systems in our evaluation.

TAC-1 (Heinzerling and Strube, 2015): This system uses local and pairwise inference in an easy-first, incremental rule-based approach. Features are based on popularity priors, contextual occurrence of keywords, entity type, and relational evidence.

TAC-2 (Sil et al., 2015): This system employs a global inference approach which partitions a document into sets of mentions that appear near each other. The partitioning is motivated by the intuition that a given mention’s immediate context provides the most salient information for disambiguation, and drastically reduces the search space during global optimization.

TAC-3 (Dai et al., 2015): This local inference system models mentions and entity context with a CNN and word embeddings.

The systems were chosen for their popularity (AIDA, SL), performance on CoNLL (FL, PH), and performance on TAC15 (TAC systems). Unless stated otherwise, we use system output provided by authors for CoNLL systems, and provided by the workshop organizers for TAC15 systems.⁹ Our evaluation does not include (Globerston et al., 2016) and (Yamada et al., 2016), who report better performance on CoNLL than PH, but were unable to make system output available.

3.1 Setup and Implementation Details

Feature extraction is implemented as a UIMA pipeline (Ferrucci and Lally, 2004); using the Stanford CoreNLP (Manning et al., 2014) UIMA components provided by DKPro (Eckart de Castilho and Gurevych, 2014) for text segmentation, POS tagging, and named entity recogni-

⁹<http://www.nist.gov/tac/2015/KBP/data.html>

⁹<http://www.nist.gov/tac/2015/KBP/data.html>

tion; DKPro WSD (Miller et al., 2013) for modeling entity mentions and links, and using Freebase (Bollacker et al., 2008) and YAGO (Hoffart et al., 2011a) as knowledge bases.

After feature extraction, we train a random forest classifier¹⁰ for each dataset, one using FL system results for the CoNLL development set (216 documents) and one using TAC-1 results for the TAC15 training set (168 documents).

For evaluation, we apply the verifier trained on FL CoNLL development results to the test set results of the FL and AIDA systems, and a verifier trained on PH training data to the PH test set results. For the test set output of TAC systems 1-3 we apply the verifier trained on the TAC15 training set output of TAC-1.

As metric we use `strong_link_match` as implemented by the Wikilinks project for the CoNLL dataset, and the official NIST scorer (Hachey et al., 2014) for TAC15. This metric measures precision, recall, and $F1$ of matching entity links and mention spans.

3.2 Results and Discussion

Evaluation results are shown in Table 5. Our method improves the linking performance of all evaluated EL systems. The impact is most noticeable for the systems that only use local and pairwise inference, namely FL (+1.9 $F1$), TAC-1 (+2.4 $F1$), TAC-3 (+1.1 $F1$). The improved TAC-1 result (68.1 $F1$) is the best published linking score on the TAC15 dataset.

Improvements are smaller for the global inference systems, AIDA, HP, and TAC-2. In contrast to Ratnov et al. (2011), who report only a very small increase in linking performance when incorporating global features into a local inference-based system, our results indicate that global features are useful and lead to considerable improvements.

As expected, improvements are caused by increased precision, due to filtering out likely linking mistakes. The fact that this increase is not accompanied by a commensurate decrease in recall, shows that our method predicts wrong linking decisions with high accuracy.

On TAC15, we observe considerable improvements in linking precision of up to 10.4 percent.

¹⁰Various other classifiers we tried, e.g. neural networks, showed no better performance during cross-validation on development sets.

Dataset	System	Baseline			After verification			Δ		
		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
CoNLL	AIDA	83.2	83.6	83.4	86.0	82.3	84.1	+2.8	-1.3	+0.7
	SPOTL	85.5	80.5	82.9	93.0	77.6	84.6	+7.5	-2.9	+1.7
	FL	85.3	85.2	85.2	89.2	84.7	86.9	+4.0	-0.5	+1.7
	PH	90.5	90.5	90.5	93.2	89.1	91.1	+2.7	-1.4	+0.6
TAC15	TAC-1	71.2	61.1	65.8	81.6	58.6	68.2	+10.4	-2.5	+2.4
	TAC-2	71.4	57.9	63.9	81.2	53.3	64.4	+9.8	-4.6	+0.5
	TAC-3	68.0	55.6	61.1	77.6	52.0	62.2	+9.6	-3.2	+1.1

Table 5: Results on CoNLL and TAC15 test sets. *Baseline* shows performance of the original systems, *After verification* shows performance after application of our automatic verification method, and Δ shows the corresponding change. Bold font indicates best results for each metric and system.

On CoNLL, the precision increase is less pronounced, arguably owing to the already higher baseline precision, which leaves less room for improvement. Since EL is usually performed as part of a larger task, such as knowledge base completion, search, or as part of a more comprehensive entity analysis system (Durrett and Klein, 2014), good precision is highly desirable in order to minimize error propagation to other system components and downstream applications.

3.3 Candidate Reranking

We resort to the binary decision of either retaining or removing an entity linked by an EL system if no candidate entities and no meaningful confidence scores are available. This is the case for the output of many EL systems, such as the systems participating in the TAC KBP TEDL 2015 challenge.

In case the EL system outputs not only the top-ranked candidate entity, but also lower-ranked ones, we can apply our verification method to all candidates and rerank them according to their probability of being correct. For example, if the EL system linked a mention to candidate entity e_1 over candidate e_2 , but verification assigns a higher probability of being correct to e_2 , we rerank e_2 over e_1 . Since we assume that the document’s semantic profile derived from EL results is sufficiently accurate, we do not recreate it after reranking a candidate.

Reranking the candidate entities produced by the FL system on the CoNLL test set, this achieves a similar increase in $F1$, but with a different precision-recall trade-off (Table 6): We observe highest precision at the cost of a lower recall for

System	Prec	Rec	F1
FL baseline	85.3	85.2	85.2
FL filter	89.2	84.7	86.9
FL rerank	87.9	85.6	86.7

Table 6: Comparison of filtering and candidate entity reranking performance on the CoNLL test set.

filtering, while reranking increases both precision and recall.

3.4 Ablation Study

We conduct an ablation study to assess the impact of the proposed global coherence features on prediction performance. Applying backward elimination (John et al., 1994), we iteratively remove one feature set and successively eliminate the feature set with the largest impact (Figure 2).

Surprisingly, the string similarity features have a large effect across all three systems. This suggests that current systems do not optimally utilize string similarity when selecting and ranking candidate entities for a given mention.

Our proposed global coherence features are among the top features for all systems. This contradicts prior findings by Ratnov et al. (2011) and shows that global coherence has a considerable impact on EL performance. We believe that this is due to our proposed coherence features being more informative than the generic semantic relatedness measures used in prior work. While ablation indeed shows a relatively low importance of semantic relatedness features (cf. SemRel in Figure 2), further research is required to test this hypothesis.

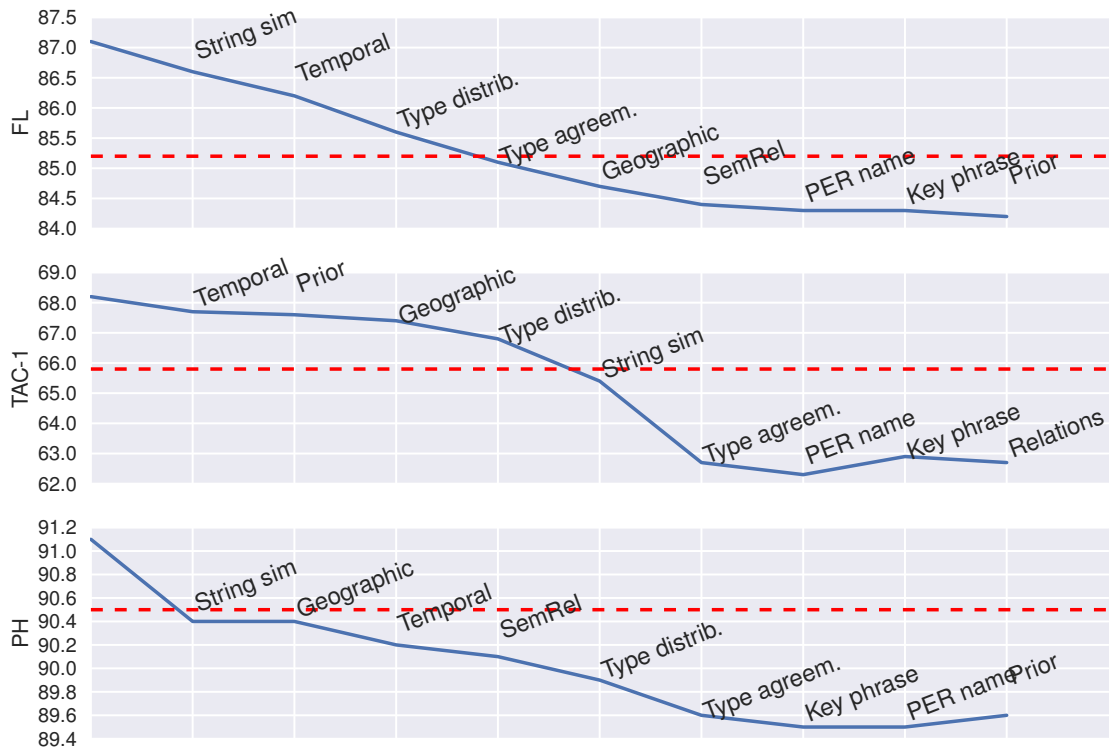


Figure 2: Feature set ablations for the FL, TAC-1, and PH systems. The solid blue lines show the performance impact in terms of `strong_link_match` $F1$ incurred from eliminating feature sets. The red dashed line indicates baseline performance without verification.

3.5 Automatic Verification on Noisy Text

The TAC15 dataset consists of different text genres: clean newswire articles, and noisy discussion forum threads. Analysis of verification performance on these two genres reveals that verification has the biggest impact on noisy text (Table 7, bottom), while the improvement is smaller for two systems on clean text, and even slightly negative for one system, namely the global inference system TAC-2 (Table 7, top).

4 Related Work

Global coherence has been successfully employed for EL in a number of seminal works (Kulkarni et al., 2009; Hoffart et al., 2011b; Han et al., 2011), and more recently by Moro et al. (2014), Pershina et al. (2015), and Globerson et al. (2016), among others. These approaches maximize global coherence based on a general notion of semantic relatedness, while considering a fixed number of candidate entities for each mentions. Our approach differs from these in in two regards. Firstly, we introduce specific aspects of coherence, namely entity type coherence, geographic coherence, and tem-

poral coherence. While these aspects are limited to certain entities, such as entities with a clearly defined location and temporal range, our experiments showed that features based on these notions of coherence are useful on the types of texts found in common datasets. Secondly, in our verification setting, these rich coherence measures can be efficiently incorporated since their computation is linear in the number of entities mentioned in a document, while they would be prohibitively expensive in the global inference EL setting.

Entity types have been used in prior work. Cucerzan (2007) maximizes the agreement of Wikipedia categories associated with candidate entities. Due to intractability of the resulting global optimization problem, the agreement of the candidate entities for a given mention is maximized with respect to all categories of all candidate entites of all other mentions, and hence includes many wrong categories. Our approach is more precise, since verification allows using only the types of the top-ranked candidate entities. Sil and Yates (2013) also employ entity types, but only maximize type agreement of entity mentions in a small context window. In contrast, our ap-

Genre	System	Baseline			After verification			Δ		
		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
News	TAC-1	66.5	60.3	63.2	75.8	57.0	65.0	9.3	-3.3	1.8
	TAC-2	69.7	59.9	64.4	79.3	53.9	64.2	9.6	-6.0	-0.2
	TAC-3	63.0	59.1	61.0	71.3	54.3	61.7	8.3	-4.8	0.7
Forum	TAC-1	76.0	61.8	68.1	87.4	60.0	71.2	11.4	-1.8	3.1
	TAC-2	73.1	56.1	63.5	83.0	52.8	64.6	9.9	-3.3	1.1
	TAC-3	73.8	52.4	61.3	84.7	49.9	62.8	10.9	-2.5	1.5

Table 7: Verification on different text genres. See caption of Table 5 for details.

proach uses global context and hence allows capturing long-distance relations.

Post-processing of EL system output has been approached as an ensembling task (Rajani and Mooney, 2016). In this setting, a meta-classifier combines the output of different EL systems on a given dataset, taking into account features such as system confidence scores, past system performance, and number of systems agreeing with a given decision. Our approach differs from ensembling, since we post-process the output of a single system, using rich semantic features. In contrast, ensembling requires multiple system outputs and relies on meta-information about system performance and decision confidence. Combining these two post-processing methods is an interesting problem for future work and could lead to further improvements, since the two methods rely on different types of information.

5 Conclusions and Future Work

We have introduced automatic verification as a post-processing step for entity linking (EL). Our method uses the output of an existing EL system to create a semantic profile of the given text using entity types, as well as geographic and temporal information. Due to the high precision achieved by state-of-the-art EL systems, this profile is a sufficiently accurate representation of the text’s main topic, and further situates the text temporally and geographically. This profile is then used to automatically verify each linked mention individually, i.e., to predict whether it has been linked correctly or not. Verification allows leveraging a rich set of global and pairwise features that would be prohibitively expensive for EL systems employing global inference. Evaluation showed consistent improvements when applying our method to seven different EL systems on two different datasets.

Our main goal in future work is the better integration of our approach with existing EL systems. Most notably, some EL systems produce meaningful confidence scores, which we currently disregard. We expect further improvements from incorporating various confidence measures into the verification process. Automatic verification could also be used in an easy-first setting to identify likely correct decisions made by a fast and simple EL system, and then perform the remaining decisions with a more sophisticated system. Since our features make use of coreference information in the form of person name agreement, as well as entity types, another line of future research is expanding our proposed entity linking verification method to entity analysis (Durrett and Klein, 2014), which models entity linking, coreference, and entity typing as a joint task.

Acknowledgments

We thank Matthew Francis-Landau, Maria Pershina, as well as the TAC KBP 2015 organizers for providing system output, and the anonymous reviewers for providing helpful feedback. This work has been supported by the German Research Foundation as part of the Research Training Group “Adaptive Preparation of Information from Heterogeneous Sources” (AIPHES) under grant No. GRK 1994/1.

References

- Elke Achtert, Ahmed Hettab, Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. 2011. Spatial outlier detection: Data, algorithms, visualizations. In *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings*, pages 512–516.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim

- Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, Vancouver, B.C., Canada, 10–12 June 2008, pages 1247–1250.
- Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, 3–7 April 2006, pages 9–16.
- Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning*, Prague, Czech Republic, 28–30 June 2007, pages 708–716.
- Hongliang Dai, Siliang Tang, Fei Wu, Zewu Ma, and Yueting Zhuang. 2015. The ZJU-EDL system for entity discovery and linking at TAC KBP 2015. In *Proceedings of the Eighth Text Analysis Conference*, Gaithersburg, MD, USA. National Institute of Standards and Technology.
- Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, pages 121–124, Graz, Austria.
- Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association of Computational Linguistics*, 2:477–490.
- Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- David A. Ferrucci and Adam Lally. 2004. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3):327–348.
- Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1256–1261, San Diego, California, June. Association for Computational Linguistics.
- Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, Berlin, Germany, August. Association for Computational Linguistics.
- Frédéric Godin, Tom De Nies, Christian Beecks, Laurens De Vocht, Wesley De Neve, Erik Mannens, Thomas Seidl, and Rik Van de Walle. 2014. The normalized freebase distance. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 218–221. Springer, Anissaras, Crete, Greece.
- Ben Hachey, Joel Nothman, and Will Radford. 2014. Cheap and easy entity evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Md., 22–27 June 2014, pages 464–469.
- M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. London, U.K.: Longman.
- Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Beijing, China, 25–29 July 2011, pages 765–774.
- Benjamin Heinzerling and Michael Strube. 2015. Visual error analysis for entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Beijing, China, 26–31 July 2015, pages 37–42.
- Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. 2011a. YAGO2: Exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th World Wide Web Conference*, Hyderabad, India, 28 March – 1 April, 2011, pages 229–232.
- Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenu, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011b. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland, U.K., 27–29 July 2011, pages 782–792.
- Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: Keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the ACM 21st Conference on Information and Knowledge Management*, Maui, Hawaii, USA, 29 October – 2 November 2010, pages 545–554.
- Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP 2015: Tri-lingual entity discovery and linking. In *Proceedings of the Eighth Text Analysis Conference*, Gaithersburg, MD,

- USA. National Institute of Standards and Technology.
- George H. John, Ron Kohavi, and Karl Pfleger. 1994. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129.
- Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of Wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 28 June – 1 July 2009, pages 457–466.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.
- Tristan Miller, Nicolai Erbs, Hans-Peter Zorn, Torsten Zesch, and Iryna Gurevych. 2013. Dkpro wsd: A generalized uima-based framework for word sense disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Sofia, Bulgaria, August. Association for Computational Linguistics.
- David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy at AAAI-08*, Chicago, Ill., 13 July 2008, pages 25–30.
- Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: A unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.
- Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, Denver, Colorado, May–June. Association for Computational Linguistics.
- Nazneen Fatema Rajani and Raymond Mooney. 2016. Combining supervised and unsupervised ensembles for knowledge base population. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1943–1948, Austin, Texas, November. Association for Computational Linguistics.
- Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Portland, Oreg., 19–24 June 2011, pages 1375–1384.
- Avirup Sil and Alexander Yates. 2013. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2369–2374, Orlando, Florida, USA. ACM.
- Avirup Sil, Giorgiana Dinu, and Radu Florian. 2015. The IBM systems for trilingual entity discovery and linking at TAC 2015. In *Proceedings of the Eighth Text Analysis Conference*, Gaithersburg, MD, USA. National Institute of Standards and Technology.
- Valentin I Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, Istanbul, Turkey, 21–27 May 2012, pages 3168–3175.
- Jannik Strötgen and Michael Gertz. 2010. Heildeltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, Uppsala, Sweden, July. Association for Computational Linguistics.
- Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 250–259, Berlin, Germany, August. Association for Computational Linguistics.

Named Entity Recognition in the Medical Domain with Constrained CRF Models

Charles Jochim

IBM Research - Ireland
Dublin, Ireland
charlesj@ie.ibm.com

Léa A. Deleris

IBM Research - Ireland
Dublin, Ireland
lea.deleris@ie.ibm.com

Abstract

This paper investigates how to improve performance on information extraction tasks by constraining and sequencing CRF-based approaches. We consider two different relation extraction tasks, both from the medical literature: dependence relations and probability statements. We explore whether adding constraints can lead to an improvement over standard CRF decoding. Results on our relation extraction tasks are promising, showing significant increases in performance from both (i) adding constraints to post-process the output of a baseline CRF, which captures “domain knowledge”, and (ii) further allowing flexibility in the application of those constraints by leveraging a binary classifier as a pre-processing step.

1 Introduction

With the number of articles indexed by MEDLINE/PubMed exceeding one million articles per year¹, manual consumption of published medical literature is no longer practical and researchers are increasingly turning to automated techniques to quickly identify and process relevant medical knowledge (e.g., literature-based discovery (Hristovski et al., 2006)). Our overall project’s objective is to semi-automate the construction of decision support models by generating probabilistic graphical models of medical conditions and their associated risks based on the academic literature (Deleris et al., 2013). An essential task in this project consists of extracting from medical papers any relations mentioning (i) dependence or independence between variables and (ii) probability

¹<https://mbr.nlm.nih.gov/Background.shtml>

statements indicating the strength of a relationship. For both types of relations, we have approached the entity extraction step as a sequence labeling problem. We then rely on a set of rules to construct relations from the entities extracted. This paper focuses specifically on the entity extraction step which we describe in more detail in the following section. We then proceed in Section 3 with details about our suggested constraint enforcement procedures. Section 4 reports details about the experiments, with numerical results reported and discussed in Section 5.

2 Dependence Relation and Probability Statement Extraction

2.1 Dependence Relation Extraction

The first task concerns identifying dependence relations between pairs of potential variables mentioned in text. Dependence and independence here are to be understood as defined by probability theory where A depends on B iff $\Pr(A) \neq \Pr(A|B)$. As independence statements seldom occur in the literature, we focus predominantly on dependence in this paper and define independence as the negation of dependence. Our choice to use the term “dependence” to describe the task may lead to some ambiguity in the NLP world yet from a probability perspective, it is a precise characterization. Thus we caution the reader that our use of the word “dependence” is exclusively related to its meaning in probability theory. We structure dependence relations as being composed of two variables (variable A and variable B) which are interchangeable, one influence term I and an optional negation.² As an example, from the sentence “For endometrial cancer, body mass index represents a major modifiable risk factor; about half of all cases in

²Negation enables us to capture independence though we ignore that last element for the remainder of the paper.

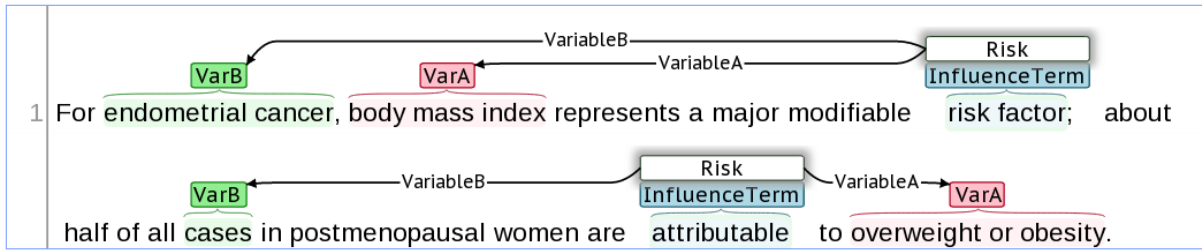


Figure 1: Dependence relation example in brat (Stenetorp et al., 2012)

postmenopausal women are attributable to overweight or obesity,” we extract two structured relations. The first involves variable A , *body mass index*, variable B , *endometrial cancer*, and an influence term, *risk factor*, and the second has a variable A , *overweight or obesity*, variable B , *cases* and influence term, *attributable* (see Figure 1).

2.2 Probability Statement Extraction

The second task focuses on extracting probability statements composed of probability terms (numbers) along with variables A (the conditioned variable) and B (the conditioning variable). Together they form a conditional probability statement, i.e., $\Pr(A|B) = x$. For instance, from the sentence “Malaysian women have a one in 20 chance of developing breast cancer in their lifetime,” we want to extract the probability number, *one in 20*; the variable A , *developing breast cancer in their lifetime*; and the variable B , *Malaysian women*, which could be represented as $\Pr(\textit{developing breast cancer in their lifetime}|\textit{Malaysian women}) = 0.05$.

2.3 Insights into the Extraction Task

Probability terms, and to a lesser extent influence terms, are fairly regular and therefore more easily classified, while risk variables A and B are heterogeneous and exhibit a broad semantic variety. “1977-1990”, “breast cancer”, “homozygous carriers”, “> or = 10 years”, and “younger than age 35” are some examples of different variables taken from our corpus.

Risk variable identification presents multiple challenges. Consider the example “Carriers of the AC haplotype, which represents the variant alleles of both SNPs, were at an increased risk (OR = 1.41, 95% CI 1.09-1.82).” We have an odds ratio probability term “OR = 1.41” and two variables. However, determining the boundaries of the variables is not straightforward. Should the condition-

ing variable be the whole subject including the relative clause, only “Carriers of the AC haplotype,” or even simply “AC haplotype”? We sidestep this issue in our current work because these variables will later be clustered and aggregated into variable groups, e.g., a variable group “breast cancer” could include mentions of “breast cancer”, “ER+”, “breast carcinoma” and others.

Finally, the distinction between variables A and B is essential when constructing a probabilistic statement (while not significant for dependence extraction) but it can be problematic to distinguish the two when extracting them from text. Overall, the variable identification task has proved quite challenging. In fact, our interest in exploring constraints is motivated by preliminary results which hinted, as we will explain in more details in the next section, that, provided a probability number or an influence term is detected, we should further encourage the algorithm to search for the other associated variables.

3 Methodology

We approach the entity extraction (probability term and variables on one side, influence terms and variables on the other side) as a sequence labeling problem. We want to identify the best sequence of labels y for a sequence of tokens x comprising a sentence. The labels in our vocabulary are O (“outside”), A (variable A), B (variable B) and, depending on the specific task, P (probability term) or I (influence term). We initially propose using conditional random fields (CRF) (Lafferty et al., 2001) for this task as they have been successful for other related NLP sequence labeling tasks (Sha and Pereira, 2003; Settles, 2004). We start with a linear-chain CRF:

$$\Pr(y|x) = \frac{1}{Z_x} \exp\left(\sum_{t=1}^T \sum_k \lambda_k f_k(y_{t-1}, y_t, x_t)\right)$$

Algorithm 1 Dependence Extraction Constraints

if influence term > 0 **then**
 ensure at least one A and one B label
else if $A > 0$ **then**
 ensure at least one B label
else if $B > 0$ **then**
 ensure at least one A label
end if

where T is the number of observations indexed by t , k indexes the feature function f_k and weight λ_k , and Z_x normalizes over the entire input sequence, $Z_x = \sum_y \exp(\sum_{t=1}^T \sum_k \theta_k f_k(y_{t-1}, y_t, x_t))$.

The CRF performs satisfactorily leading to higher precision classification of the labels than recall. However, it is not able to capture some information that we know to be true. For instance, for these corpora, if there is a labeled influence or probability term, 92% and 99% of the time, respectively, there are co-occurring variables in the same sentence. Thus for probability statement extraction, if we have a sentence with a detected probability term, which can be reliably classified (with F_1 scores around 74), then we want to enforce the presence of both variables A and B . B is theoretically optional, as it is possible to find marginal probability, i.e., statements with an empty conditioning set. In practice, given our domain, we choose to impose the presence of at least one B label for each sentence with a P label. By contrast, for a sentence without a probability term, we want to discard such output altogether. In the case of dependence statement extraction, the situation is similar, if we detect an influence term or a variable, we want to force the system to find the other relevant pieces of the relations. The constraints we apply to each of our two tasks are summarized in structured language in Algorithm 1 and Algorithm 2.

While it is straightforward to discard sentences that do not contain a given label, constraining the output with statements to enforce the presence of at least one type of label given the presence of another label, is less obvious. We address it by complementing the initial classification with further steps to enforce the constraints. We have previously explored different approaches, varying in complexity, for finding the most likely sequence while applying these constraints (Deleris and Jochim, 2016). Here we settle on the constrained approach inspired by Culotta and McCal-

Algorithm 2 Probability Extraction Constraints

if probability term = 0 **then**
 ensure no labeled entities
else
 ensure at least one A and one B label
end if

lum (2004) that uses posterior conditional probability in the CRF decoding.

3.1 Notation

We denote by y a vector of labels associated with observations x (tokens). Let T be the number of observations (x_1, \dots, x_T) then y also contains T elements (y_1, \dots, y_T) which we index by t and $y_t \in \{A, B, P, I, O\} \forall t = 1 : T$.

Let $y^* = \operatorname{argmax}_y \Pr(y|x)$ denote the output of applying Viterbi decoding to our observations x . To describe the proposed extensions, we then introduce the following variables : $t_P = \{t \in [1 : T] : y_t^* = P\}$, the indexes in the initial output corresponding to the label P . $t_I = \{t \in [1 : T] : y_t^* = I\}$, the indexes in the initial output corresponding to the label I . $t_A = \{t \in [1 : T] : y_t^* = A\}$, indexes corresponding to the label A . $t_B = \{t \in [1 : T] : y_t^* = B\}$, indexes corresponding to the label B . Finally we denote by t_O the set of unspecified indexes, i.e., $t_O = \{t \in [1 : T] : y_t^* = O\}$.

Our **baseline** method is simply to evaluate the classifier performance based on y^* . As we mentioned above, our specific context for probability statement extraction leads us to discard all labels in a sentence that does not contain any P label. We thus implement this constraint in our baseline as a simple post-processing filter on the CRF output. Specifically, for probability statement extraction, for a sentence such that $|t_P| = 0$, then we define a modified output y^B (B standing here for Baseline) where $y_t^B = O \forall t = 1 : T$, else we keep the original output so that $y_t^B = y_t^* \forall t = 1 : T$.

3.2 CRF-Driven Constraints

Our chosen method to enforce constraints takes into account current knowledge (i.e., initial decoding from CRF) when estimating probabilities of labels by conditioning posterior probabilities on the presence of label A and B based on current observations. Specifically, if only A missing, i.e.,

$|t_A| = 0$ then we search

$$t_A^* = \operatorname{argmax}_{t \in t_O} \varphi_t(s) \quad (1)$$

and we define $y_t^C = A \forall t \in t_A^*$ and $y_t^C = y_t^B \forall t \notin t_A^*$.

In this method, φ represents the the posterior *conditional* probability given the observed locations of the required labels. In the case of probability statement extraction, we have: $\varphi_t(s) = \Pr(y_t = s | x, y_{t_P} = P, y_{t_A} = A, y_{t_B} = B)$ for $s \in \{A, B, P, O\}$. In the case of dependence relation extraction, we have: $\varphi_t(s) = \Pr(y_t = s | x, y_{t_I} = I, y_{t_A} = A, y_{t_B} = B)$ for $s \in \{A, B, I, O\}$.

Similar procedures are applied when B is missing and when both labels are missing where we search jointly on the best locations for A and B .

One difficulty with this approach is the need to compute φ . For this purpose, we borrow from Culotta and McCallum (2004) who provide a method to compute the forward values $\alpha_t(s) = \Pr(x_1, \dots, x_t, y_t = s)$ of the forward-backward algorithm when forced to conform to a subpath of constraints $C = \langle s_t, s_{t+1}, \dots \rangle$. These constraints specify for a subset of locations which states they must be in or not be in (negative constraints).

The original recursive approach to compute $\alpha_t(s)$ is

$$\alpha_{t+1}(s) = \sum_{s'} \alpha_t(s') \exp \left(\sum_k \lambda_k f_k(s', s, x_{t+1}) \right) \quad (2)$$

The updated recursion equation proposed by Culotta and McCallum (2004) so as to compute $\alpha'_t(s) = \Pr(x_1, \dots, x_t, y_t = s | C)$ is simply to apply Equation 2 when $y_{t+1} = s$ conforms with C (including locations that are not constrained in any way in C) and set $\alpha'_{t+1}(s) = 0$ otherwise. Note that in (Culotta and McCallum, 2004), the indexes of the constraints included in C are assumed to be contiguous although the method also applies when they are not. In our case, for probability statement extraction, we will simply set $C = \{y_{t_P} = P, y_{t_A} = A, y_{t_B} = B\}$, again where at least one of the sets t_A or t_B is empty.

We similarly extend this method to compute constrained backward values $\beta'_t(s) = \Pr(x_{t+1}, \dots, x_T | y_t = s, C)$ by proposing the modified backward recursion

$$\beta'_t(s) = \sum_{s'} \beta'_{t+1}(s') \exp \left(\sum_k \lambda_k f_k(s, s', x_{t+1}) \right) \quad (3)$$

when $y_t = s$ conforms with C and set $\beta'_{t+1}(s) = 0$ otherwise. Overall, this means that $\varphi_t(s) = \Pr(y_t = s | x, y_{t_P} = P, y_{t_A} = A, y_{t_B} = B)$ can be expressed as follows:

$$\varphi_t(s) = \frac{\Pr(y_t = s, x | y_{t_P} = P, y_{t_A} = A, y_{t_B} = B)}{\Pr(x | y_{t_P} = P, y_{t_A} = A, y_{t_B} = B)} \quad (4)$$

This is the result of an application of Bayes rule along with the conditional independence assumptions of the CRF. In turn, we have that :

$$\varphi_t(s) = \frac{\alpha'_t(s) \beta'_t(s)}{\sum_{s'} \alpha'_T(s')} \quad (5)$$

The output y^C is guaranteed to contain at least one label A and one label B . t_A^* in Equation 1 will contain only one token while variables A and B , in reality, often span multiple tokens. Therefore, as an additional post-processing step, we run the Viterbi algorithm once more using the identified labels A, B and P as constraints. This may reveal longer spans with such labels.

3.3 Classifier-driven Constraints

As we report in Section 5, imposing constraints based on the initial CRF decoding improves recall more than it degrades precision and thus proves useful. We further explore whether adding flexibility to the process can help reduce the precision degradation. Indeed in the case of dependence relations, we observed that several influence terms are quite common in dependence relations but are still not found exclusively in these relations. In our dataset for instance, the most common influence term words are *associated* (142 occurrences in the training set), *association* (42), *risk* (36), and *increased* (22). Our one-step CRF-based approach may thus be misled by those common words into forcing the presence of a dependence relation.

Therefore, we introduce two separate binary classifiers to predict whether or not the sentence contains either a dependence relation or a probability statement. Our intuition is that the classifier will be a more reliable indicator of the presence of a relation than the entity extraction of either probability number or influence term. We make use of that prediction to determine if and how to apply the constraints. In fact, we apply a threshold on the confidence of the classifier in order to decide whether to enforce the constraints.

To coordinate the classification, the entity detection (baseline CRF) and the constraint enforce-

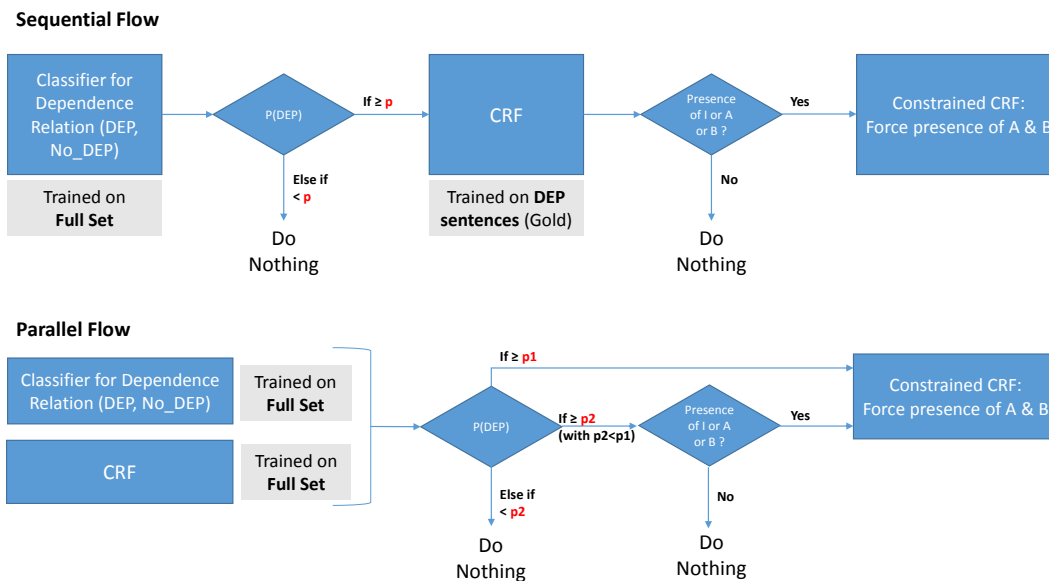


Figure 2: Description of Constraint Enforcement Flows

ment steps, we explore two kinds of flows as depicted in Figure 2 for dependence relation extraction. Both flows start with the binary classifier but evolve differently:

- In the *sequential flow*, if the confidence of the classifier is below a threshold p we simply discard the sentence. If it is above the threshold, then we process the sentence through the baseline CRF trained only on sentences containing relations in the ground truth. We then enforce our constraints, i.e., for the dependence case, discarding the sentence if no variable or influence term has been found but forcing the presence of variables if we detect an influence term or at least one variable.
- In the *parallel flow*, we implement the baseline CRF in parallel with the binary classifier. This implies that the baseline CRF is trained on the full set of sentences. Afterwards, if the confidence of the classifier is below p^2 , we discard the sentence, if it is above p^1 we enforce our constraints regardless of the Baseline CRF output. For intermediate cases, we only enforce constraints if we detect either an influence term or a variable.

4 Experiments

In this section we first describe the data used in our experiments and then cover the configuration

of the baseline CRF classifier along with the constrained configurations we test.

Data for Dependence Statements. The dependence dataset comes from 210 abstracts selected from PubMed based on a query about breast cancer. These 210 abstracts are split into 2144 sentences, of which 785 have a dependence relation. The dependence relations include 830 variables labeled A and 837 labeled B . The annotation also includes labels for *influence terms*, *modifiers* and *negation* (as mentioned in Section 2.1). The latter two are not considered here as they do not contribute to the dependence relations, but we do show results for influence terms as they are useful in constructing the dependence relations (although that is not covered in this paper).

Data for Probability Statements. The probability dataset is similarly constructed with 194 abstracts from PubMed that are related to breast cancer. The whole dataset has 2078 sentences, 376 of which, contain probabilistic statements, for example, “81% p of stage III/IV breast cancers B were positive for SNCG expression A .” These sentences contain 652 probability terms, 446 variables A , and 467 variables B .

We split both datasets into train (70%) and test (30%) sets. Parameter tuning can be done by splitting the training set, however experiments shown in this paper do not require parameter tuning.

Baseline CRF We initially evaluate a baseline CRF model without constraints, implemented with Mallet (McCallum, 2002). The same feature set used in the baseline is used throughout our experiments. It is composed of standard features for sequence tagging: surface form, lemma, POS tag, word shape (i.e., is capitalized, has digit, etc.), and the arc label from a dependency parse. The CRF also extracts features from the previous position ($t - 1$) and the following position ($t + 1$) as well as bigram features combining the previous two positions ($t - 2, t - 1$). We do not tune the features or regularization parameters in our experiments, but instead focus on the differences from applying constraints in decoding. The CRF training is the same for nearly all experiments and the important changes are in decoding and in how the constraints are applied, so tuning these parameters should not affect our results. The one exception to this is the sequential flow where the CRF classifier is trained only on sentences with relations. We use the default value for the Gaussian variance prior (10) and keep the same features across our experiments.

In addition to the baseline CRF we test three constraint settings: (i) **Default** refers to the decoding with the CRF driven enforcement of constraints described in Section 3.2; (ii) **Parallel** refers to the application of the Parallel flow which feeds the sentences into the CRF to identify entities and a binary classifier to determine if the sentence has a relation. The output of the binary classifier determines if and how the decoding constraints should be applied; and (iii) **Sequential** refers to the application of the sequential flow where the binary classifier first filters out noisy sentences followed by the use of a CRF trained on relation sentence data.

For the binary classifier, we make use of IBM’s Natural Language Classifier service³ which is relies on a Convolutional Neural Network combined with word embeddings (Feng et al., 2015). To clarify, the classifiers that we use, while using pre-trained word embedding (on general domain), are then only trained with our own training data.

Evaluation criteria. We evaluate our approach using standard metrics: precision, recall, and F_1 . In addition, we consider different criteria for matching entities, i.e., token matches, exact entity matches, and “sloppy” matches (Olsson et al.,

2002). In Section 5 we report only our results for token matching since the trends are consistent for different matching criteria. Exact entity matching is unnecessarily strict for our application (e.g., in the context of our work “AC haplotype” is just as good as “Carriers of the AC haplotype”) and measuring performance by token makes the results easily interpretable.

5 Results

5.1 Binary Classifier

As mentioned in Section 3.3, we introduce a binary classifier to decide whether constraints should be applied or not (instead of relying only on the entity annotation of variables A , B , and Influence or Probability terms). We need an accurate classifier to ensure that constraints are only applied when necessary. Our experiments show that the classifier achieves good results with accuracy for classifying dependence sentences of 82.2% and for probability statements, 95.3%.

5.2 Entity Extraction for Dependence Relations

We first look at the entity extraction results for dependence relations in Table 1. There is consistent improvement in F_1 scores for variables A and B as we refine the application of constraints. The baseline CRF classifier has modest precision but much weaker recall and misses a number of variable A and B entities entirely. We are not particularly concerned about boundary errors with the CRF and they are relatively infrequent. On the other hand, the missed entities, i.e., the entity *false negatives*, are more detrimental to the results. In fact, of the 606 sentences in the test set, 253 should have some dependence relation annotation but 143 of these are missing a variable A , 143 are missing a variable B , and 84 are missing both. This result motivates our use of domain constraints.

Adding Default constraints leads to an increase in F_1 as recall improves while precision drops. When the baseline CRF assigns any A , B , or influence term, we force it to have both variables A and B , and essentially by forcing it we recover enough new variables (higher recall) to offset making some less confident prediction (lower precision). The application of these constraints depends on the quality of the Baseline predictions for A , B , and Influence Term, and we note that that

³<https://www.ibm.com/watson/developercloud/nl-classifier.html>

	Precision	Recall	$F_{\beta=1}$
VarA			
Baseline	60.27%	34.26%	43.69
Default	55.34%	45.68%	50.05
Parallel	64.16%	44.95%	52.87
Sequential	56.86%	62.19%	59.41
VarB			
Baseline	41.53%	24.54%	30.85
Default	43.19%	40.44%	41.77
Parallel	52.15%	40.11%	45.35
Sequential	46.78%	48.67%	47.71
Influence Term			
Baseline	68.45%	42.95%	52.78
Default	67.18%	43.96%	53.14
Parallel	65.10%	55.70%	60.04
Sequential	64.86%	56.38%	60.32

Table 1: Entity extraction results for dependence relations.

even for influence terms our performance is moderate and inferior to that of the binary classifier.

The Parallel results display improvements to both precision and recall over the baseline. While Parallel recall for A and B is slightly below the numbers reached in Default, the 9% absolute increase in precision for both variables A and B leads to F_1 improvements for both. By applying the binary classifier prediction before applying constraints, we remove some noisy sentences that contain spuriously labeled entities, which are not in a dependence relation; in the Parallel results there are 56 such sentences. For example, in the baseline experiment a lone variable B , *subsequent cancer occurrence*, is extracted from the sentence “It allowed for a correct estimation of the risk, and for investigating the time trend of the subsequent cancer occurrence.” The classifier correctly filters out this sentence and prevents the CRF from classifying what might be a valid variable B had there actually been a dependence relation.

Naturally, by filtering out sentences we also risk discarding some with dependence relations and thus affecting recall. In one example of this negative case, the baseline experiment labels only an influence term without variables A or B . The constraints applied for the Default constraint experiment assign the (correct) variable B and (incorrect) variable A . However, the classifier in the Par-

allel experiment mistakenly leads us not to label any entities in this sentence, thereby missing the relation.

We observe similar patterns for the Sequential results. The overall best F_1 scores for variables A and B are due to the significant increases in recall. The main difference between Parallel and Sequential is the fact that the CRF is trained on a filtered set of sentences for Sequential compared to the full set for Parallel. The Sequential CRF, using filtered data, has a higher ratio of variables labeled A and B per word seen in training and this makes it more confident in predicting labels A and B in testing. This helps the Sequential flow recover several new entities which are missed by the baseline CRF. This also explains the higher recall but lower precision with respect to Parallel.

Although only variables A and B are necessary for dependence relations, we also report performance on influence terms, which contribute to the identification of dependence relations. For the influence terms, we observe that precision decreases with each experiment as recall increases. The binary classifier has a strong and positive effect on influence term extraction as we see an increase in recall and F_1 . Recall for Parallel and Sequential increases by about 11% and 15% (absolute).

Table 1 shows Parallel results with $p^1 = 0.5$ and $p^2 = 0.5$ and Sequential results with $p = 0.5$ (see Figure 2). We chose those parameter values as they correspond to the default for a binary classifier. In Section 5.4 we look more closely at the effects of p , p^1 and p^2 .

5.3 Entity Extraction for Probability Statement

The probability results reported in Table 2 display similar trends as those for dependence relations, though we note a few differences as well. First the Baseline results are lower for variables A and B . This is due mainly to the heterogeneity of these variables (even with respect to the dependence relations) and to the smaller dataset. To illustrate how much more heterogeneous the variables are, the type-token ratio for variable A is 0.188 in the dependence training set and 0.392 in the probability training set, and the difference for variable B is similar, 0.224 vs. 0.448. The number of sentences in the dependence and probability corpora are similar but probability statements are less frequent than dependence relations and for the 620

	Precision	Recall	$F_{\beta=1}$
VarA			
Baseline	33.98%	8.93%	14.14
Default	52.81%	31.12%	39.17
Parallel	53.41%	33.93%	41.50
Sequential	57.41%	46.43%	51.34
VarB			
Baseline	38.61%	11.40%	17.61
Default	43.27%	26.32%	32.73
Parallel	39.01%	25.44%	30.80
Sequential	47.73%	42.98%	45.23
Probability Term			
Baseline	79.43%	69.24%	73.99
Default	79.65%	70.17%	74.61
Parallel	81.56%	69.71%	75.17
Sequential	80.36%	75.89%	78.06

Table 2: Entity extraction results for probability statements.

sentences in the test set we only have 127 with probability annotation. Like with dependence relations, for probability statements we can motivate our constraints by looking at the large number of missed (i.e., false negative) entities. The baseline CRF misses variable A in 94 sentences with probability statements (i.e., about 74% of probability statements are missing variable A). There are 78 probability sentences with a missed variable B , and 57 missing both.

By applying constraints to these missing variables we observe the largest increase in performance occurs from the Baseline to the Default setting, with improvements in precision, recall, and F_1 . The Default constraints approach hinges on the CRF prediction of the probability term, which performs well. Because this prediction is already reliable, by contrast with the dependence relation extraction, there is less potential for improvement by applying the binary classifier. In fact, for Parallel, performance increases for variable A but decreases for variable B .

Training the CRF solely on probability statements, as is the case in Sequential, appears to have a greater impact than with dependence. As the proportion of sentences with no-relation in the probability dataset is higher than in the dependence dataset, filtering out the no-relations sentences removes more noise in the case of prob-

p	Dependence		Probability	
	VarA	VarB	VarA	VarB
0.00	48.94	36.23	51.10	44.70
0.05	57.01	46.69	52.40	47.22
0.25	59.77	47.05	51.25	46.90
0.50	59.41	47.71	51.34	45.23
0.75	59.41	49.42	51.28	45.55
0.95	58.72	49.52	46.37	39.79
1.00	0.00	0.00	0.00	0.00

Table 3: F_1 scores across p threshold values for Sequential flow.

ability statement extraction. In turn, removing this noise improves the CRF performance, leading in particular to the improved performance for the probability term and consequently to improvements in the variable extraction with constraints.

5.4 Improving Performance through Threshold Values

Performance was earlier reported for the Parallel and Sequential flow for $p = p^1 = p^2 = 0.5$, which represents the default threshold for binary classifiers. In this section, we look closer at the effect of these threshold values on performance.

There are no values for p that consistently lead to maximum F_1 scores (Tables 3–7) and there is not space to show precision and recall results as well. However, as would be expected, the precision goes up as p^1 and p^2 increase, i.e., as we become more conservative in constraining the CRF output. On the other hand, recall drops with higher values of the p threshold as the CRF is not pushed to find previously missed variables A or B .

We do find lower p^2 thresholds perform better for probability than dependence. This is likely because the probability constraints based on the CRF are still quite reliable. The classifier for dependence statements must be more strict in filtering sentences, using higher p^2 thresholds, because the dependence constraints are less reliable.

The Sequential flow results are similar. The maximum F_1 scores on the probability dataset come with $p = 0.05$, like p^2 values for Parallel. The p threshold for variables A and B varies, also similar to p^2 for Parallel, with better A results coming from a lower threshold and better B results with a higher threshold. However, more work needs to be done to see how we can best leverage classification in the Sequential and Parallel flows.

p^2 p^1	0.00	0.05	0.25	0.50	0.75	0.95
0.00	46.48	–	–	–	–	–
0.05	50.81	52.64	–	–	–	–
0.25	50.87	52.34	53.88	–	–	–
0.50	51.26	52.77	53.93	52.87	–	–
0.75	51.18	52.69	53.85	52.34	52.30	–
0.95	51.17	52.66	53.83	52.32	51.82	51.33
1.00	50.05	51.53	52.70	51.10	50.55	49.67

Table 4: Var. A F_1 for Dependence Parallel flow.

p^2 p^1	0.00	0.05	0.25	0.50	0.75	0.95
0.00	36.98	–	–	–	–	–
0.05	42.42	44.93	–	–	–	–
0.25	43.38	44.81	45.34	–	–	–
0.50	43.36	44.80	44.61	45.35	–	–
0.75	43.10	44.53	44.33	44.67	45.45	–
0.95	43.18	44.61	44.41	44.50	44.79	44.78
1.00	41.77	43.13	42.86	42.91	43.18	42.66

Table 5: Var. B F_1 for Dependence Parallel flow.

6 Related Work

Our work touches on several areas from constrained conditional models (Goldwasser et al., 2012) to biomedical entity extraction. The work most related to our approach for applying constraints with CRF decoding is (Culotta and McCallum, 2004; Roth and Yih, 2005; Kristjansson et al., 2004). Our solution for constraining CRF decoding borrows from Culotta and McCallum (2004) (which is also used by Kristjansson et al. (2004)). They use constraints for calculating the ‘forward’ values in the forward-backward algorithm and use this for estimating confidence at given states in the sequence. Our work applies constraints both forward and backward and uses these global constraints to force specific entities to be extracted. Roth and Yih (2005) also apply constraints to the CRF but instead of using Viterbi decoding as is done here they use Integer Linear Programming (ILP) to add constraints in decoding for semantic role labeling.

With respect to our overall objective of entity and relation extraction from the medical literature, a large proportion of the related work originates from BioNLP event extraction (Kim et al., 2009; Nédellec et al., 2013; Chaix et al., 2016; Deléger et al., 2016). These tasks are similar in that they

p^2 p^1	0.00	0.05	0.25	0.50	0.75	0.95
0.00	18.88	–	–	–	–	–
0.05	42.50	42.75	–	–	–	–
0.25	41.01	41.22	41.35	–	–	–
0.50	41.38	41.60	41.73	41.50	–	–
0.75	40.90	41.11	41.24	41.00	41.01	–
0.95	40.74	40.94	41.07	40.83	40.84	34.49
1.00	39.17	39.34	39.47	39.20	39.19	32.35

Table 6: Var. A F_1 for Probability Parallel flow.

p^2 p^1	0.00	0.05	0.25	0.50	0.75	0.95
0.00	13.90	–	–	–	–	–
0.05	31.39	31.58	–	–	–	–
0.25	31.31	31.51	31.27	–	–	–
0.50	31.85	32.06	31.82	30.80	–	–
0.75	31.96	32.17	31.93	30.91	30.88	–
0.95	31.87	32.09	31.84	30.80	30.77	21.78
1.00	32.73	32.96	32.71	31.64	31.62	22.31

Table 7: Var. B F_1 for Probability Parallel flow.

extract biomedical entities, analogous to our variables A and B , and the relations between entities (e.g., bio-molecular events).

The most similar entity extraction task to ours is from Fiszman et al. (2007). They are interested in extracting mentions of diseases and medical risk factors⁴ from medical literature. They take a less statistical and more semantic approach to convert the biomedical text into a semantic representation using the UMLS Semantic Network.

7 Conclusions

In this paper we investigate how we can improve performance on information extraction tasks by constraining CRF-based approaches. We investigate two relation extraction tasks from the medical literature – dependence relations and probability statements – and show that by using our constrained CRF models we can get significant improvements over a CRF baseline. In future work we plan to build on these improvements and test constraints jointly applied to entity and relation extraction to improve our project’s construction of decision support models.

⁴Their *risks* and *disorders* appear to be subsets of variables A and B .

References

- Estelle Chaix, Bertrand Dubreucq, Abdelhak Fatihi, Dialekti Valsamou, Robert Bossy, Mouhamadou Ba, Louise Deléger, Pierre Zweigenbaum, Philippe Bessières, Loïc Lepiniec, and Claire Nédellec. 2016. Overview of the regulatory network of plant seed development (seedev) task at the bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 1–11, Berlin, Germany, August. Association for Computational Linguistics.
- Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 109–112, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.
- Louise Deléger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferré, Philippe Bessières, and Claire Nédellec. 2016. Overview of the bacteria biotope task at bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 12–22, Berlin, Germany, August. Association for Computational Linguistics.
- Léa A. Deleris and Charles Jochim. 2016. Probability statements extraction with constrained conditional random fields. In *Proceedings of MIE2016*, pages 527–531.
- Léa Amandine Deleris, Bogdan Sacaleanu, and Lamia Tounsi. 2013. Extracting risk modeling information from medical articles. In *MEDINFO 2013 - Proceedings of the 14th World Congress on Medical and Health Informatics, 20-13 August 2013, Copenhagen, Denmark*, page 1158.
- Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 813–820.
- Marcelo Fiszman, Graciela Roseblat, Caroline B. Ahlers, and Thomas C. Rindfleisch. 2007. Identifying risk factors for metabolic syndrome in biomedical text. In *Proceedings of the AMIA Annual Symposium*, pages 249–253.
- Dan Goldwasser, Vivek Srikumar, and Dan Roth. 2012. Predicting structures in nlp: Constrained conditional models and integer linear programming in nlp. In *NAACL HLT 2012 Tutorial Abstracts*, Montréal, Canada, June. Association for Computational Linguistics.
- Dimitar Hristovski, Carol Friedman, Thomas C. Rindfleisch, and Borut Peterlin. 2006. Exploiting semantic relations for literature-based discovery. In *AMIA Annual Symposium Proceedings*, pages 349–353.
- Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado, June. Association for Computational Linguistics.
- Trausti Kristjansson, Aron Culotta, Paul Viola, and Andrew McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *Proceedings of the 19th National Conference on Artificial Intelligence, AAAI'04*, pages 412–418. AAAI Press.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Fredrik Olsson, Gunnar Eriksson, Kristofer Franzén, Lars Asker, and Per Lidén. 2002. Notions of correctness when evaluating protein name taggers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, COLING '02*, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Dan Roth and Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of the 22Nd International Conference on Machine Learning, ICML '05*, pages 736–743, New York, NY, USA. ACM.
- Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 107–110, Geneva, Switzerland, August 28th and 29th. COLING.
- Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, NAACL '03*, pages 134–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April. Association for Computational Linguistics.

Learning and Knowledge Transfer with Memory Networks for Machine Comprehension

Mohit Yadav **Lovekesh Vig** **Gautam Shroff**
TCS Research New-Delhi TCS Research New-Delhi TCS Research New-Delhi
y.mohit@tcs.com lovekesh.vig@tcs.com gautam.shroff@tcs.com

Abstract

Enabling machines to read and comprehend unstructured text remains an unfulfilled goal for NLP research. Recent research efforts on the “machine comprehension” task have managed to achieve close to ideal performance on simulated data. However, achieving similar levels of performance on small real world datasets has proved difficult; major challenges stem from the large vocabulary size, complex grammar, and the frequent ambiguities in linguistic structure. On the other hand, the requirement of human generated annotations for training, in order to ensure a sufficiently diverse set of questions is prohibitively expensive. Motivated by these practical issues, we propose a novel curriculum inspired training procedure for Memory Networks to improve the performance for machine comprehension with relatively small volumes of training data. Additionally, we explore various training regimes for Memory Networks to allow knowledge transfer from a closely related domain having larger volumes of labelled data. We also suggest the use of a loss function to incorporate the asymmetric nature of knowledge transfer. Our experiments demonstrate improvements on Dailymail, CNN, and MCTest datasets.

1 Introduction

A long-standing goal of NLP is to imbue machines with the ability to comprehend text and answer natural language questions. The goal is still distant and yet generates tremendous amount of interest due to the large number of potential NLP applications that are currently stymied because of

their inability to deal with unstructured text. Also, the next generation of search engines are aiming to provide precise and semantically relevant answers in response to questions-as-queries; similar to the functionality of digital assistants like *Cortana* and *Siri*. This will require text understanding at a non-superficial level, in addition to reasoning, and, making complex inferences about the text.

As pointed out by Weston et al. (2016), the Question Answering (QA) task on unstructured text is a sound benchmark on which to evaluate machine comprehension. The authors also introduced *bAbI*: a simulation dataset for QA with multiple toy tasks. These toy tasks require a machine to perform simple induction, deduction, multiple chaining of facts, and, complex reasoning; which make them a sound benchmark to measure progress towards AI-complete QA (Weston et al., 2016). The recently proposed Memory Network architecture and its variants have achieved close to ideal performance, i.e., more than 95% accuracy on 16 out of a total of 20 QA tasks (Sukhbaatar et al., 2015; Weston et al., 2016).

While this performance is impressive, and is indicative of the memory network having sufficient capacity for the machine comprehension task, the performance does not translate to real world text (Hill et al., 2016). Challenges in real-world datasets stem from the much larger vocabulary, the complex grammar, and the often ambiguous linguistic structure; all of which further impede high levels of generalization performance, especially with small datasets. For instance, the empirical results reported by Hill et al. (2016) show that an end-to-end memory network with a single hop surpasses the performance achieved using multiple hops (i.e, higher capacity), when the model is trained with a simple heuristic. Similarly, Tapaswi et al. (2015) show that a memory network heavily overfits on the MovieQA dataset and

yields near random performance. These results suggest that achieving good performance may not always be merely a matter of training high capacity models with large volumes of data. In addition to exploring new models there is a pressing need for innovative training methods, especially when dealing with real world sparsely labelled datasets.

With the advent of deep learning, the state of art performance for various semantic NLP tasks has seen a significant boost (Collobert and Weston, 2008). However, most of these techniques are *data-hungry*, and require a large number of sufficiently diverse labeled training samples, e.g., for QA, training samples should not only encompass an entire range of possible questions but also have them in sufficient quantity (Bordes et al., 2015). Generating annotations for training deep models requires a tremendous amount of manual effort and is often too expensive. Hence, it is necessary to develop effective techniques to exploit data from a related domain in order to reduce dependence on annotations. Recently, Memory Networks have been successfully applied to QA and dialogue-systems to work with a variety of disparate data sources such as movies, images, structured, and, unstructured text (Weston et al., 2016; Weston, 2016; Tapaswi et al., 2015; Bordes et al., 2015). Inspired from the recent success of Memory Networks, we study methods to train memory networks with small datasets by allowing for knowledge transfer from related domains where labelled data is more abundantly available.

The focus of this paper is to improve generalization performance of memory networks via an improved learning procedure for small real-world datasets and knowledge transfer from a related domain. In the process, this paper makes the following major contributions:

- (i) A curriculum inspired training procedure for memory network is introduced, which yields superior performance with smaller datasets.
- (ii) The exploration of knowledge transfer methods such as pre-training, joint-training and the proposed curriculum joint-training with a related domain having abundant labeled data.
- (iii) A modified loss function for joint-training to incorporate the asymmetric nature of knowledge transfer, and also investigate the application of a pre-trained memory network on very small datasets such as MCTest dataset.

The remainder of the paper is organized as follows: Firstly, we provide a summary of related work in Section 2. Next in Section 3, we describe the machine comprehension task and the datasets utilized in our experiments. An introduction to memory networks for machine comprehension is presented in Section 4. Section 5 outlines the proposed methods for learning and knowledge transfer. Experimental details are provided in Section 6. We summarize our conclusions in Section 7.

2 Related Work

Memory Networks have been successfully applied to a broad range of NLP and machine learning tasks. These tasks include but are not limited to: performing reasoning over a simulated environment for QA (Weston et al., 2016), factoid and non-factoid based QA using both knowledge bases and unstructured text (Kumar et al., 2015; Hill et al., 2016; Chandar et al., 2016; Bordes et al., 2015), goal driven dialog (Bordes and Weston, 2016; Dodge et al., 2016; Weston, 2016), automatic story comprehension from both video and text (Tapaswi et al., 2015), and, transferring knowledge from one knowledge-base while learning to answer questions on a different knowledge base (Bordes et al., 2015). Recently, various other attention based neural models (similar to Memory Networks) have been proposed to tackle the machine comprehension task by QA from unstructured text (Kadlec et al., 2016; Sordoni et al., 2016; Chen et al., 2016). To the best of our knowledge, knowledge transfer from an unstructured text dataset to another unstructured text dataset for machine comprehension is not explored yet.

Training deep networks is known to be a notoriously hard problem and often the success of these techniques hinges upon achieving higher generalization performance with high capacity models (Blundell et al., 2015; Larochelle et al., 2009; Glorot and Bengio, 2010). To address this issue, *Curriculum learning* was firstly introduced by Bengio et al. (2009), which showed that training with gradually increasing difficulty leads to a better local minima, specially when working with non-convex loss functions. Although devising a universal curriculum strategy is hard, as even humans do not converge to one particular order in which concepts should be introduced (Rohde and Plaut, 1999) some notion of concept difficulty is normally utilized. With similar motivations, this pa-

per makes an attempt to exploit *curriculum learning* for machine comprehension with a memory network. Recently, curriculum learning has also been utilized to avoid negative transfer and make use of task relatedness for multi-task learning (Lee et al., 2016). Concurrently, Sachan and Xing (2016) have also studied curriculum learning for QA and unlike this paper, they do not consider learning and knowledge transfer on small real-world machine comprehension dataset in the setting of memory networks.

Pre-training & word2vec: Pre-training can often mitigate the issue that comes with random initialization used for network weights, by guiding the optimization process towards the basins of better local minima (Mishkin and Matas, 2016; Krahenbuhl et al., 2016; Erhan et al., 2010). An inspiration from the ripples created by the success of pre-training and as well as word2vec, this paper explores pre-training to utilize data from a related domain and also pre-trained vectors from word2vec tool (Mikolov et al., 2013). However, finding an optimal dimension for these pre-trained vectors and other involved hyper-parameters requires computationally extensive experiments.

Joint-training / Co-training / Multi-task learning / Domain adaptation: Previously, the utilization of common structures and similarities across different tasks / domains has been instrumental for various closely related learning tasks refereed as joint-training, co-training, multi-task learning and domain adaptation (Collobert and Weston, 2008; Liu et al., 2015; Chen et al., 2011; Maurer et al., 2016). To mitigate this ambiguity, in this paper, we limit ourselves to using “joint-training” and refrain from co-training, as unlike this work, co-training was initially introduced to exploit unlabelled data in the presence of small labelled data and two different and complementary views about the instances (Blum and Mitchell, 1998).

While this work looks conceptually similar, the proposed method tries to exploit information from a related domain and aims to achieve an asymmetric transfer only towards the specified domain, without any interest in the source domain, and hence should not be confused with the long-standing pioneering work on multi-task learning (Caruana, 1997). Another field of work that is related to this paper is on domain adaptation which appears to have two major related branches. The first branch is the recent work that has primar-

ily focused on unsupervised domain adaptation (Nguyen and Grishman, 2015; Zhang et al., 2015), and the other is the traditional work on domain adaptation which has focussed on problems like entity recognition and not on machine comprehension and modern neural architectures (Ben-David et al., 2010; Daume III, 2007).

3 Machine Comprehension : Datasets and Tasks Description

Machine comprehension is the ability to read and comprehend text, i.e., understand its meaning, and can be evaluated by tasks involving the answering of questions posed on a context document. Formally, a set of tuples (q, C, S, s) is provided, where q is the question, C is the context document, S is a list of possible answers, and, s indicates the correct answer. Each of q , C , and S are sequence or words from a vocabulary V . Our aim is to train a memory network model to perform QA with small training datasets. We propose two primary ways to achieve this: 1) Improve the learning procedure to obtain better models, and 2) Demonstrate knowledge transfer from a related domain.

3.1 Data Description

Several corpora have been introduced for the machine comprehension task such as MCTest-160, MCTest-500, CNN, Dailymail, and, Children Boot Test (CBT) (Richardson et al., 2013; Hermann et al., 2015; Hill et al., 2016). The MCTest-160 and MCTest-500 have multiple-choice questions with associated narrative stories. Answers in these datasets can be one of these forms: a word, a phrase, or, a full sentence.

The remaining datasets are generated using Cloze-style questions; which are created by deleting a word from a sentence and asking the model to predict the deleted word. A place-holder token is substituted in place of the deleted word which is also the correct answer (Hermann et al., 2015). We have created three subsets of CNN namely, CNN-11K, CNN-22K and CNN-55K from the entire CNN dataset, and Dailymail-55K from the Dailymail dataset. Statistics on the number of samples comprising these datasets is presented in Table 1.

3.2 Improve Learning Procedure

It has been shown in the context of language modelling that presenting the training samples in an easy to hard ordering allows for shielding

	MCTest-160	MCTest-500	CNN-11K	CNN-22K	CNN-55K	Dailymail-55K
# Train	280	1400	11,000	22,000	55,000	55,000
# Validation	120	200	3,924	3,924	3,924	2,500
# Test	200	400	3,198	3,198	3,198	2,000
# Vocabulary	2856	4279	26,550	31,932	40,833	42,311
# Words \notin Dailymail-55K	—	—	1,981	2,734	6,468	—

Table 1: Number of samples in training, validation, and, test samples in the MCTest-160, MCTest-500, CNN-11K, CNN-22K, CNN-55K, and, Dailymail-55K datasets; along with the size of vocabulary.

the model from very hard samples during training, yielding faster convergence and better models (Bengio et al., 2009). We investigate a curriculum learning inspired training procedure for memory networks to improve performance on the three subsets of the CNN dataset described below.

3.3 Demonstrate Knowledge Transfer

We plan to demonstrate knowledge transfer from Dailymail-55K to three subsets of CNN of varying sizes utilizing the proposed join-training method. For learning, we make use of smaller subsets of the CNN dataset. The smaller size of these subsets enables us to assess the performance boost due to knowledge transfer: As our aim is to demonstrate transfer when less labelled data is available, choosing the complete dataset would render gains from knowledge transfer as insignificant. We also demonstrate knowledge transfer for the case of MCTest dataset using embeddings obtained after training the memory network with CNN datasets.

4 End-to-end Memory Network for Machine Comprehension

End-to-end Memory Network is a recently introduced neural network model that can be trained in an end-to-end fashion; directly on the tuples (q, C, S, s) using standard back-propagation (Sukhbaatar et al., 2015). The complete training procedure can be described in the three steps: i) encoding the training tuples into the contextual memory, ii) attending context in memory to retrieve relevant information with respect to a question, and, iii) predicting the answer using the retrieved information. To accomplish the first step, an embedding matrix $A \in \mathbb{R}^{p \times d}$ is used to map both question and context into a p -dimensional embedding space; by applying the following transformations: $\vec{q} = A\Phi(q)$ and $\{\vec{m}_i = A\Phi(c_i)\}_{i=1,2,\dots,n}$. Where n is the number of items in context C and Φ is a bag-of-words representation in d -dimensional space, where d is typically the size of the vocabulary V . In the

second step, the network senses relevant information present in the memory \vec{m}_i for query \vec{q} , by computing the attention distribution $\{\alpha_i\}_{i=1,2,\dots,n}$, where $\alpha_i = \text{softmax}(\vec{m}_i^T \vec{q})$. Thereafter, α_i is used to aggregate the retrieved information into a vector representation \vec{r}_o by utilizing another memory \vec{r}_i ; as stated in Equation 1. The memory representation \vec{r}_i is also defined as $\{\vec{r}_i = B\Phi(c_i)\}_{i=1,2,\dots,n}$ in a manner similar to \vec{m}_i using another embedding matrix $B \in \mathbb{R}^{p \times d}$.

$$\vec{r}_o = \sum_{i=1}^n \alpha_i \vec{r}_i \quad (1)$$

$$\hat{a}_i = \text{softmax}((\vec{r}_o + \vec{q})^T U\Phi(s_i)) \quad (2)$$

In the last step, prediction distribution \hat{a}_i is computed as in Equation 2, where $U \in \mathbb{R}^{p \times d}$ is an embedding matrix similar to A and can potentially be tied with A , and s_i is one of the answers in S . Using the prediction step, a probability distribution \hat{a}_i over all s_i can be obtained and the final answer is selected as the one with the highest probability \hat{a}_i corresponding to the option s_i .

$$L(P, D) = \frac{1}{N_D} \sum_{n=1}^{N_D} a_n \times \log(\hat{a}_n(P, D)) + (1 - a_n) \times \log(1 - \hat{a}_n(P, D)) \quad (3)$$

To train a memory network, the cross-entropy loss function L between the true label distribution $a_i \in \{0, 1\}^s$ (which is a one hot vector to indicate the correct label s in the training tuples) and the predicted distribution \hat{a}_i is used, as in Equation 3. Where P , D and N_D represent the set of model parameters to learn, training dataset, and the number of tuples in the training set respectively. Such an objective can be easily optimized using stochastic gradient descent (SGD). A memory network can easily be extended to perform several hops over the memory before predicting the answer. For details, we refer to Hill et al. (2016). However, we constrain this study to use a single-hop network in order to reduce number of

parameters to learn and also the chances of over-fitting; as we are dealing with small scale datasets.

Self-Supervision is a heuristic introduced to provide memory supervision and the rationale behind is that if the memory supporting the correct answer is retrieved than the model is more likely to predict the correct answer (Hill et al., 2016). More precisely, this is achieved by keeping a hard attention over memory while training, i.e., $m'_o = \text{argmax } \alpha_i$. At each step of SGD, the model computes m'_o and updates only using those examples which do not select the memory m'_o having the correct answer in the corresponding c_i .

5 Proposed Methods

We attempt to improve the training procedure for Memory Networks in order to increase the performance for machine comprehension by QA with small scale datasets. Firstly, we introduce an improved training procedure for memory networks using curriculum learning which is termed as *Curriculum Inspired Training* (CIT) and offer details about this in Section 5.1. Thereafter, Section 5.2 explains joint-training method for knowledge transfer from an abundantly labelled dataset to another dataset with limited label information .

5.1 CIT: Curriculum Inspired Training

Curriculum learning makes use of the fact that model performance can be significantly improved if the training samples are not presented randomly but in such a way so as to make the learning task gradually more difficult by presenting examples in an easy to hard ordering (Bengio et al., 2009). Such a training procedure allows the learner to waste less time with noisy or hard to predict data when the model is not ready to incorporate such samples. However, what remains unanswered and is left as a matter of further exploration is how to devise an effective strategy for a given task?

$$SF(q, S, C, s) = \frac{\sum_{\text{word} \in \{q \cup S \cup C\}} \log(\text{Freq.}(\text{word}))}{\#\{q \cup S \cup C\}} \quad (4)$$

In this work, we formulate a curriculum strategy to train a memory network for machine comprehension. Formally, we rank training tuples (q, S, C, s) from easy to hard based on the normalized word frequency for passage, question, and context initially; using the score function (SF) mentioned in Equation 4 (i.e. easier passages have

more frequent words). The training data is then divided into a fixed number of chapters, with each successive chapter resulting in addition of more difficult tuples. The model is then trained sequentially on each chapter with the final chapter containing the complete training data. The presence of both the number of chapters and the fixed number of epochs per chapter makes such a strategy flexible and allows to be tailored to different data after optimizing the like other hyper-parameters.

$$L(P, D, en) = \frac{1}{N_D} \sum_{n=1}^{N_D} (a_n \times \log(\hat{a}_n(P, D)) + (1 - a_n) \times \log(1 - \hat{a}_n(P, D))) \times \mathbf{1}(en, c(n) \times epc) \quad (5)$$

The loss function used for curriculum inspired training varies with epoch number; as mentioned in Equation 5. Note, in Equation 5, en and $c(n)$ represents the current epoch number and chapter number for n^{th} tuple assigned using rank allocated based on SF mentioned in Equation 4 respectively. epc , P , D , and $\mathbf{1}$ is the number of epochs per chapter, model parameters, training set, and an indicator function which is one if first argument is \geq the second argument or else zero; respectively.

5.2 Joint-Training for Knowledge Transfer

While joint-training methods offer knowledge transfer by exploiting similarities and regularities across different tasks or datasets, the asymmetric nature of transfer and skewed proportion of datasets is usually not handled in a sound way. Here, we devise a training loss function \hat{L} to relieve both of these involved issues while doing joint-training with a target dataset (TD) with fewer training samples and a source dataset (SD) having label information for higher number of examples; as mentioned in Equation 6.

$$\hat{L}(P, TD, SD) = 2 \times \gamma \times L(P, TD) + 2 \times (1 - \gamma) \times L(P, SD) \times F(N_{TD}, N_{SD}) \quad (6)$$

Where \hat{L} represents the devised loss function for joint-training for transfer, L the cross-entropy loss function also mentioned earlier in Equation 3, γ is a weighting factor which varies between zero and one, $F(N_{TD}, N_{SD})$ is an another weighting factor which is a function of number of samples in the target domain N_{TD} and in the source domain N_{SD} . The rationale behind γ factor is to control the relative update in the network due to

samples from source and target datasets; which permits biasing of the model performance towards one dataset. $F(N_{TD}, N_{SD})$ factor can be independently utilized to mitigate the effect of skewed proportion in the number of samples present in both target and source domains. Note, maintaining both γ and $F(N_{TD}, N_{SD})$ as separate parameters allows for restricting γ within (0,1) without any extra computation as described below.

5.3 Improved Loss Functions

This paper explores the following variants of the introduced loss function \hat{L} for knowledge transfer via joint-training:

1. Joint-training (Jo-Train):- $\gamma = 1/2$ and $F(N_{TD}, N_{SD}) = 1$.
2. Weighted joint-training (W+Jo-Train):- $\gamma = (0, 1)$ and $F(N_{TD}, N_{SD}) = N_{TD}/N_{SD}$.
3. Curriculum joint-training (CIT+Jo-Train):- $L(P, TD)$ & $L(P, SD)$ of Equation 6 are replaced by their analogous terms $L(P, TD, en)$ & $L(P, SD, en)$ generated using Equation 5; $\gamma = 1/2$ and $F(N_{TD}, N_{SD}) = 1$.
4. Weighted curriculum joint-training (W+CIT+Jo-Train):- $L(P, TD)$ & $L(P, SD)$ of Equation 6 are replaced by analogous $L(P, TD, en)$ & $L(P, SD, en)$ generated using Equation 5; $\gamma = (0, 1)$ and $F(N_{TD}, N_{SD}) = N_{TD}/N_{SD}$.
5. Source only (SrcOnly) :- $\gamma = 0$.

The $F(N_{TD}, N_{SD})$ factor does not increase computation as it is not optimized for any of the cases. Jo-Train (Liu et al., 2015), SrcOnly and a method similar to W+Jo-Train (Daume III, 2007) have also been explored previously for other NLP tasks and models.

6 Experiments

We evaluate the performance on datasets introduced earlier in Section 3. We first present baseline methods, pre-processing and training details. In Section 6.3, we present results on CNN-11/22/55K, MCTest-160 and MCTest-50 to validate our claims mentioned in Section 1. All of the methods presented here are implemented in Theano (Bastien et al., 2012) and Lasagne (Dieleman et al., 2015) and are run on a single GPU (Tesla K40c) server with 500GB of memory.

6.1 Baseline Methods

We implemented Sliding Window (SW) and Sliding Window + Distance (SW+D)(Richardson et al., 2013) as baselines to compare against our experiments. Further, we augment SW (or SW+D) to incorporate distances between word vectors of the question and the context over the sliding window; in a manner similar to the way SW+D is augmented from SW by Richardson et al. (2013). These approaches are named based upon the source of pre-trained word vectors, e.g., SW+D+CNN-11K+W2V utilizes vectors estimated from both CNN-11K and word2vec pre-trained vectors¹. In case of more than one source, individual distances are summed and utilized for final scoring. Results on MCTest for SW, SW+D, and their augmented approaches are reported using online available scores for all answers².

Meaningful Comparisons: To ascertain that the improvement is due to the proposed training methods, and not merely because of addition of more data, we built multiple baselines, namely, initialization using word vectors from word2vec, pre-training, Jo-train, and SrcOnly. For pre-training and word2vec, words \in target dataset and \notin source dataset are initialized, by a uniform random sampling with the limits set to the extremes spanned by the word vectors in the source domain. *It is worth to note that the pre-training and Jo-train utilizes as much label information and data as other proposed variants of joint-training. Also, SrcOnly method is an indicative of how much direct knowledge transfer from source domain to target domain can be achieved without any learning.*

6.2 Pre-processing & Training Details

While processing data, we replace words occurring less than 5 times by <unk> token except for MCTest datasets. Additionally, all entities are included in vocabulary. All models are trained by carrying out the optimization using SGD with learning rate in $\{10^{-4}, 10^{-3}\}$, momentum value set to 0.9, weight decay in $\{10^{-5}, 10^{-4}\}$, and, max norm in $\{1, 10, 40\}$. We kept length of window equal to 5 for CNN / Dailymail datasets(Hill et al., 2016) and for MCTest datasets is chosen from $\{3, 5, 8, 10, 12\}$. For embedding size, we look for the optimal value in $\{50, 100, 150, 200, 300\}$ for

¹<http://code.google.com/p/word2vec>

²<http://research.microsoft.com/en-us/um/redmond/projects/mctest/results.htm>

Model + Training Methods	CNN-11 K			CNN-22 K			CNN-55 K		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
SW §	21.33	20.35	21.48	21.80	20.61	20.76	21.54	19.87	20.66
SW+D §	25.45	25.40	25.90	25.61	25.25	26.47	25.85	25.74	26.94
SW+W2V §	43.90	43.01	42.60	45.70	44.10	42.23	45.06	44.50	43.50
MemNN §	98.98	45.96	46.08	98.07	49.28	51.42	97.31	54.98	56.69
MemNN+CIT §	96.44	47.17	49.04	98.36	52.43	52.73	91.14	57.26	57.68
SW+Dailymail ‡	30.19	31.21	30.60	31.70	30.87	32.01	31.56	33.07	31.08
MemNN+W2V ‡	86.57	43.78	45.99	94.1	49.98	51.06	95.2	51.47	53.66
MemNN+SrcOnly ‡	25.12	26.78	27.08	25.43	26.78	27.08	24.79	26.78	27.08
MemNN+Pre-train ‡	92.82	52.87	52.06	95.12	53.59	55.35	96.33	56.64	59.19
MemNN+Jo-train ‡	65.78	53.85	55.06	64.85	55.94	55.69	77.32	57.76	57.99
MemNN+CIT+Jo-train ‡	77.74	55.93	55.74	78.96	55.98	56.85	71.89	56.83	59.07
MemNN+W+Jo-train ‡	71.72	54.30	55.70	79.64	55.91	56.73	71.15	57.62	58.34
MemNN+W+CIT+Jo-train ‡	80.14	56.91	57.02	79.04	57.90	57.71	76.91	58.14	59.88

Table 2: Train, validation and test percentage accuracy on CNN-11/22/55K datasets. § and ‡ indicate that the data used comes from either of CNN-11/22/55K and also from Dailymail-55K along with either of CNN-11/22/55K respectively. Random test accuracy on these datasets is 3.96% approximately.

CNN / Dailymail datasets. For CNN / Dailymail, we have trained memory network using a single batch with self-supervision heuristic (Hill et al., 2016). In case of curriculum learning, the number of chapters are optimized out of {3, 5, 8, 10} and number of epochs per chapter is set equal to $\frac{2M}{M+1} \times \frac{ed_{ncl}}{ed_{cl}} \times EN$ which is estimated by equating to the number of network update found for the optimal case of non-curriculum learning. Here M and ed_{cl} represents the number of chapter and embedding size for curriculum learning, and ed_{ncl} & EN represents the optimal value found for embedding size and number of epochs without curriculum learning. We use early stopping with a validation set while training the network.

6.3 Results & Discussion

In this section, we present results to validate contributions mentioned in Section 1. Table 2 presents the results of our approaches along with results from baseline methods SW, SW+D, SW+W2V, and a standard memory network (MemNN). Results for CIT on CNN-11/22/55K (MemNN+CIT) show an absolute improvement of 2.96%, 1.31%, and, 1.00% respectively, when compared with the memory network (MemNN) (*contribution (i)*). Figure 1 shows that the CIT leads to better convergence when compared without CIT on CNN-11K.

As baselines for knowledge transfer from the Dailymail-55K dataset to CNN-11/22/55K datasets, Table 2 presents results for SW+Dailymail, memory network initialized with word2vec (MemNN+W2V), memory network trained on Dailymail (MemNN+SrcOnly), memory network initialized with pre-trained

embeddings from Dailymail (MemNN+Pre-train) and memory network jointly-trained with both Dailymail and CNN (MemNN+Jo-train) (*contribution (ii)*). Further, results show the knowledge transfer observed when MemNN+CIT+Jo-train and MemNN+W+Jo-Train are utilized to train Dailymail-55K with CNN-11/22/55K. On combining the MemNN+CIT+Jo-train with MemNN+W+Jo-Train (which is MemNN+W+CIT+Jo-Train), a significant and consistent improvement can be observed; as the performance goes up by 1.96%, 2.03%, and, 1.89% on CNN-11/22/55K respectively; when compared against the other competitive baselines (*contribution (ii) & (iii)*).

Results empirically support the major premise of this study, i.e., CIT and knowledge transfer from a related dataset with memory network can significantly improve the performance; improvements of 10.94%, 6.28%, and, 3.19% are observed with CNN-11/22/55K respectively when compared with the standard memory network. The improvement in knowledge transfer decreases as the amount of data in the target domain starts increasing from 11K to 55K, as the volume of data in the target domain starts becoming comparable to source domain, and is enough to achieve similar level of performance without knowledge transfer.

Previously, Chen et al. (2016) annotated a sample of 100 questions on CNN stories based on the type of capabilities required to answer the question. We report results for all 6 specific categories in Table 3. Even with CNN-11K and Dailymail-55K which is roughly 20% of the complete CNN dataset, the proposed methods achieve similar per-

Model + Training Methods	Exact	Para.	Part.Clue	Multi.Sent.	Co-ref.	Ambi./Hard
SW §	3(23.1%)	12(29.2%)	2(10.5%)	0(0.0%)	0(0.0%)	2(11.7%)
SW+D §	6(46.1%)	14(34.1%)	2(10.5%)	0(0.0%)	0(0.0%)	3(17.6%)
SW+W2V §	10(76.9%)	20(48.7%)	5(26.3%)	0(0.0%)	0(0.0%)	7(41.1%)
MemNN §	8(61.5%)	20(48.7%)	12(63.1%)	1(50.0%)	0(0.0%)	2(11.7%)
MemNN+CIT §	10(76.9%)	19(46.3%)	12(63.1%)	1(50.0%)	3(37.5%)	2(11.7%)
SW+Dailymail ‡	6(46.1%)	19(46.3%)	5(26.3%)	0(0.0%)	0(0.0%)	2(11.7%)
MemNN+W2V ‡	6(46.1%)	27(65.8%)	5(26.3%)	0(0.0%)	0(0.0%)	7(41.1%)
MemNN+SrcOnly §	6(46.1%)	12(29.2%)	2(10.5%)	0(0.0%)	0(0.0%)	2(11.7%)
MemNN+Pre-train ‡	11(84.6%)	25(60.9%)	12(63.1%)	0(0.0%)	0(0.0%)	1(5.9%)
MemNN+Jo-train ‡	8(61.5%)	29(70.7%)	10(52.6%)	2(100%)	0(0.0%)	5(29.4%)
MemNN+CIT+Jo-train ‡	10(76.9%)	27(65.8%)	10(52.6%)	0(0.0%)	3(37.5%)	5(29.4%)
MemNN+W+Jo-train ‡	11(84.6%)	29(70.7%)	10(52.6%)	2(100%)	0(0.0%)	5(29.4%)
MemNN+W+CIT+Jo-train ‡	11(84.6%)	27(65.8%)	10(52.6%)	2(100%)	3(37.5%)	5(29.4%)
Chen et al. (2016) §	13(100%)	39(95.1%)	17(89.5%)	1(50.0%)	3(37.5%)	1(5.9%)
Sordoni et al. (2016) §	13(100%)	39(95.1%)	16(84.2%)	1(50.0%)	3(37.5%)	5(29.4%)
Total Number Of Samples	13	41	19	2	8	17

Table 3: Question-specific category analysis of percentage test accuracy with only learning and knowledge transfer methods on CNN-11K dataset. § and ‡ indicates that the data used comes from CNN-11K and from Dailymail-55K along with CNN-11K respectively. § indicate results from Sordoni et al. (2016).

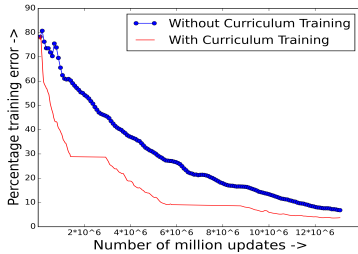


Figure 1: Percentage training error v/s number of million updates while training on CNN-11K with or without curriculum inspired training.

Training Methods	MCTest-160			MCTest-500		
	One	Multi.	All	One	Multi.	All
SW	66.07	53.12	59.16	54.77	53.04	53.83
SW+D	75.89	60.15	67.50	63.23	57.01	59.83
SW+D+W2V	79.46	59.37	68.75	65.07	58.84	61.67
SW+D+CNN-11K	79.78	59.37	67.67	64.33	57.92	60.83
SW+D+CNN-22K	76.78	60.93	68.33	64.70	59.45	61.83
SW+D+CNN-55K	78.57	59.37	68.33	65.07	59.75	62.16
SW+D+CNN-11K+W2V	77.67	59.41	68.69	65.07	61.28	63.00
SW+D+CNN-22K+W2V	78.57	60.16	69.51	66.91	60.00	63.13
SW+D+CNN-55K+W2V	79.78	60.93	70.51	66.91	60.67	63.50

Table 4: Knowledge transfer results on MCTest-160 and MCTest-500 datasets. One and Multi. indicates the questions that require one and multiple supporting facts. Random test accuracy is 25% here, as number of options are 4.

formance on 4 out of 6 categories, when compared to latest models (2^{nd} & 3^{rd} last rows of Table 3).

On very small datasets such as MCTest-160 and MCTest-500, it is not feasible to train memory network (Smith et al., 2015), therefore, we explore the use of word vectors from the embedding matrix of a model pre-trained on CNN datasets. Here, the embedding matrix refers to the encoding matrix A used in the first step of memory network as mentioned in Section 4. SW+D+CNN-11/22/55K are the results when the similarity measures comes from SW+D as mentioned in Section 6.1 and also using the word vectors from encoding matrix A obtained after training on CNN-11/22/55K. From table 4, it is evident that performance improves as the amount of data increases in CNN domain (*contribution(iii)*). Further, on combining with word2vec distance (SW+D+CNN-11/22/55K+W2V), an improvement is observed.

7 Conclusion

Looking at the widespread applications of Memory Networks and the prohibitive data requirements for training them, this paper seeks to improve the performance of memory networks on small datasets in two different ways. Firstly, this paper introduces an effective CIT procedure for machine comprehension. Secondly, this paper explores various methods to exploit labelled data from closely related domains; in order to perform knowledge transfer and improve performance. Additionally, this paper suggests the use of a modified loss function to further incorporate the asymmetric nature of knowledge transfer. Beyond machine comprehension, we believe that the proposed methods are likely to achieve higher generalization for other tasks utilizing memory network style architectures, by virtue of the proposed CIT method and joint-training for knowledge transfer.

References

- Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning Neural Information Processing Systems (NIPS) Workshop.
- Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1):151–175.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 41–48. ACM.
- Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.
- Antonie Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1606.03126*.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.
- Rich Caruana. 1997. Multitask learning. *Mach. Learn.*, 28(1):41–75, July.
- Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. 2016. Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*.
- Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. *Advances in Neural Information Processing Systems (NIPS)*, pages 2456–2464.
- Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167, New York, NY, USA. ACM.
- Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.
- Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, J Kelly, et al. 2015. Lasagne: First release. *Zenodo: Geneva, Switzerland*.
- Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, 11:625–660.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children’s books with explicit memory representations. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.
- Philipp Krahenbuhl, Carl Doersch, Jeff Donahue, and Trevor Darrell. 2016. Data-dependent initializations of convolutional neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.
- Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. 2009. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research (JMLR)*, 10:1–40.

- Giwoong Lee, Eunho Yang, and Sung Ju Hwang. 2016. Asymmetric multi-task learning based on task relatedness and loss. In *Proceedings of the 33rd Annual International Conference on Machine Learning (ICML)*, pages 230–238.
- Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, Denver, Colorado, May–June. Association for Computational Linguistics.
- Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. 2016. The benefit of multitask representation learning. *Journal of Machine Learning Research (JMLR)*, 17(81):1–32.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.
- Dmytro Mishkin and Jiri Matas. 2016. All you need is a good init. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China, July. Association for Computational Linguistics.
- Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Douglas L.T. Rohde and David C. Plaut. 1999. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72(1):67–109.
- Mrinmaya Sachan and Eric P. Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1693–1698, Lisbon, Portugal, September. Association for Computational Linguistics.
- Alessandro Sordani, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. arXiv preprint arXiv:1606.02245.
- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Advances in Neural Information Processing Systems (NIPS)*, pages 2440–2448.
- Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Movieqa: Understanding stories in movies through question-answering. arXiv preprint arXiv:1512.02902.
- Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *Proceedings of International Conference on Learning Representations (ICLR)*.
- Jason Weston. 2016. Dialog-based language learning. arXiv preprint arXiv:1604.06045.
- Xu Zhang, Felix X. Yu, Shih-Fu Chang, and Shengjin Wang. 2015. Deep transfer network: Unsupervised domain adaptation. arXiv preprint arXiv:1503.00591.

If No Media Were Allowed inside the Venue, Was Anybody Allowed?

Zahra Sarabi and Eduardo Blanco

Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203

zahrasarabi@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents a framework to understand negation in positive terms. Specifically, we extract positive meaning from negation when the negation cue syntactically modifies a noun or adjective. Our approach is grounded on generating potential positive interpretations automatically, and then scoring them. Experimental results show that interpretations scored high can be reliably identified.

1 Introduction

Negation is a complex phenomenon present in all human languages, allowing for the uniquely human capacities of denial, contradiction, misrepresentation, lying, and irony (Horn and Wansing, 2015). Acquiring and understanding negation poses unique challenges. For example, children acquire negation after learning to communicate in positive terms (Nordmeyer and Frank, 2013), and adults take longer to process sentences containing negation (Clark and Chase, 1972).

In any given language, humans communicate in positive terms most of the time, and use negation to express something unusual or an exception (Horn, 1989). But negation is ubiquitous (Morante and Sporleder, 2012): In scientific papers, 13.76% of sentences contain a negation (Szarvas et al., 2008); in product reviews, 19% (Councill et al., 2010); and in Conan Doyle stories, 22.23% (Morante and Daelemans, 2012).

From a theoretical perspective, it is accepted that negation has scope and focus, and that humans intuitively understand positive meanings from negation (Rooth, 1992; Huddleston and Pullum, 2002). For example, from (1) *John didn't earn a steady paycheck until he was 40 years old*, humans understand that (1a) *John earned unsteady paychecks before he was 40 years old*, and that

(1b) *John earned steady paychecks when he was 40 years old*. This kind of positive interpretations would benefit language understanding in general. For example, a question answering system would benefit from interpretation (1b) when answering question *Did John ever earn a steady paycheck?*

Within computational linguistics, automated approaches to extract positive meanings from negation target verbal negation (Section 3), i.e., when the negation cue is grammatically associated with a verb, as in (1). Verbal negation accounts only for a portion of all negations, e.g., out of all syntactic dependencies indicating a negation modifier (*neg* dependency) in OntoNotes (Hovy et al., 2006), 64.4% modify verbs, 19.6% nouns, 10.3% adjectives, and 5.7% other part-of-speech tags. Non-verbal negation also conveys positive meanings, e.g., from (2) *No media were allowed inside the venue* (*No* modifies noun *media*), humans understand that (2a) *Somebody* (e.g., *invited guests*) *were allowed inside the venue* and that (2b) *Media were allowed somewhere outside the venue* (presumably in a designated press area). Similarly, from (3) *She was not alive when she got to the Lafayette area* (*not* modifies adjective *alive*), humans understand that (3a) *She was dead when she got to the Lafayette area* and that (3b) *She was alive before she got to the Lafayette area*.

This paper presents new corpora and experimental results to extract positive interpretations from negation when the negation cue modifies a noun or adjective. The main contributions are: (1) analysis of negation in OntoNotes beyond verbal negation; (2) procedure to automatically generate potential positive interpretations from non-verbal negation, specifically, when the negation cue modifies a noun or adjective; (3) annotations validating and scoring potential interpretations according to their likelihood;¹ and (4) experimental results showing that the task can be automated.

¹Available at <http://www.cse.unt.edu/~blanco/>

2 Terminology and Background

Negation can be expressed by verbs (e.g., *avoid* the highway), nouns (e.g. *lack* of knowledge), adjectives (e.g., it is *useless*), adverbs (e.g., John *never* drives on the highway), and others (van der Wouden, 1997). The primary negative prefixes in English are *in-*, *il-*, *im-*, *ir-*, *un-*, *non-*, *anti-* and *a-* (Garner, 2009, p. 563). We refer to the token, prefix or suffix that indicates negation (emphasized in the examples above) as *negation cue*. As we shall see, in this paper we target negations whose cue syntactically modifies a noun or adjective.

In philosophy and linguistics, it is generally accepted that negation conveys positive meanings (Horn, 1989). We use the term *positive interpretation* to refer to the positive meaning intuitively understood by humans when reading sentences that contain negation. Positive interpretations range from implicatures (Blackburn, 2008), to entailments. *Potential* positive interpretations are positive interpretations whose validity is unknown.

Scope and Focus. Negation is generally understood in terms of scope and focus. Scope is “the part of the meaning that is negated” and focus is “the part of the scope that is most prominently or explicitly negated” (Huddleston and Pullum, 2002). Scope and focus are not exclusive of negation. Among many others, there has been work on detecting the scope of uncertainty cues (Farkas et al., 2010), and other focus-sensitive phenomena include adverbs and conditionals (Rooth, 1985).

Consider statement (2) again, *No media were allowed inside the venue*. By definition, scope refers to “all elements whose individual falsity would make the negated statement strictly true”, and focus is “the element of the scope that is intended to be interpreted as false to make the overall negative true” (Huddleston and Pullum, 2002). If any of the truth conditions below were false, statement (2) would be true, thus the scope of the negation marked with *No* is (2a-2c):

- 2a. Some people were allowed somewhere.
- 2b. Media were allowed somewhere.
- 2c. Some people were allowed inside the venue.

Choosing the element of the scope which is the focus is more challenging than identifying the scope. A natural reading of statement (2) indicates that there were people (e.g., invited guests) allowed inside the venue, and that media were (probably) allowed in a press area outside (but not far from) the venue. The former positive inter-

pretation corresponds to choosing *Media* as focus, and the latter corresponds to choosing *inside* as focus. Statement (2) exemplifies two core properties of the work presented here: We choose several foci for a single negation, and as a result, reveal several positive interpretations per negation. Further, we see positive interpretations as probabilistic knowledge, i.e., knowledge that may be likely but not necessarily certain.

In general, the task of identifying foci and revealing positive interpretations is natural to humans, but hard to automate. Consider modified statement (2') *No media were allowed inside the venue to record the presentation*. The scope is conditions (2a-2c) and (2d) *Somebody was allowed to record the presentation*. The positive interpretations from (2') are different than from (2), e.g., *Media were allowed inside the venue, but they weren't allowed to record the presentation* can only be extracted from (2').

3 Previous Work

Within computational linguistics, most approaches to process negation target scope or focus detection. Generally speaking, there are corpora with scope annotations for all types of negations, but corpora with focus annotations are restricted to verbal negation, i.e., when the negation cue is grammatically associated with a verb.

Scope of Negation. There are two main corpora with scope of negation annotations: BioScope in the medical domain (Szarvas et al., 2008) and CD-SCO (Morante and Daelemans, 2012). The annotations schemas differ substantially; CD-SCO annotates negation cues, their scopes, and the negated events or properties. There have been several supervised proposals to detect the scope of negation using BioScope and CD-SCO (Morante and Daelemans, 2009; Velldal et al., 2012; Basile et al., 2012). Fancellu et al. (2016) present the best results to date with CD-SCO using neural networks. They also perform out-of-domain evaluation with new annotations on Wikipedia, and analyze the main sources of errors.

Outside BioScope and CD-SCO, Reitan et al. (2015) present a scope detector for negation in tweets, and use it for sentiment analysis.

As the examples throughout this paper show (e.g., Section 2), detecting the scope of negation is insufficient to reveal the positive interpretations we target in this work.

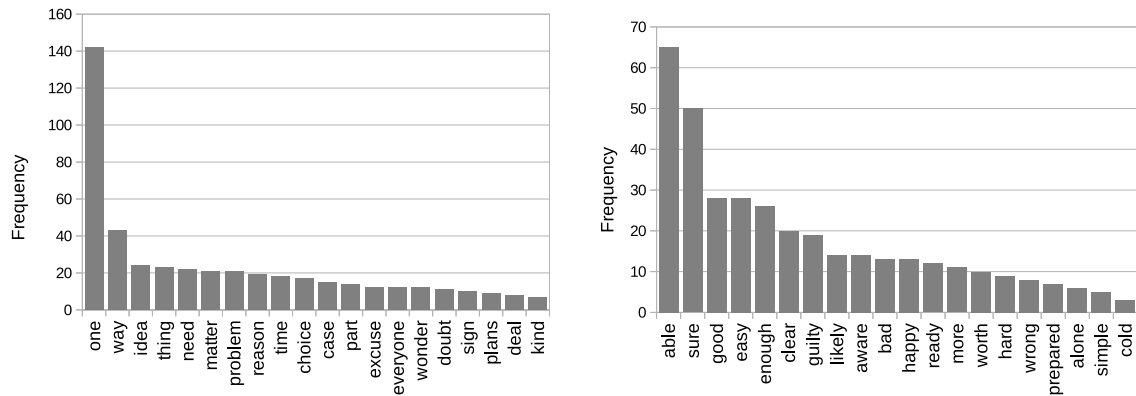


Figure 1: Most frequent nouns (left) and adjectives (right) tokens that are negated (*neg* dependency) in OntoNotes. Total number of noun and adjective tokens modified by a negation cue is 1,866 and 979.

Focus of Negation and Positive Interpretations.

Identifying the focus of negation is equivalent to revealing positive interpretations—everything but the focus is actually positive. The definition of focus does not specify annotation guidelines, and most existing efforts are grounded on semantic roles. Blanco and Moldovan (2011) annotate focus on the negations marked with ARG-M-NEG role in PropBank (Palmer et al., 2005). They select a single focus per negation, specifically, they select the role that reveals the “most useful [positive] information.” Anand and Martell (2012) refine these annotations and differentiate positive interpretations arising from focus identification, scalar implicature and neg-raising predicates. Blanco and Sarabi (2016) propose a similar approach that scores the likelihood of several potential foci per negation. The main limitations of all these previous works is that selecting as focus a semantic role is only suitable when the negation cue modifies a predicate, and roles often yield coarse-grained interpretations. Sarabi and Blanco (2016) bypass these drawbacks by working with syntactic dependencies to refine coarse-grained interpretations.

All these previous efforts to reveal positive interpretations from negation target exclusively verbal negation, i.e., when the negation cue modifies a verb. While verbal negation is more frequent (64.4% of *neg* dependencies in OntoNotes, Section 4), in this paper we target two understudied yet important negations: when the negation cue modifies a noun or adjective (19.6% and 10.3% of *neg* dependencies). Our approach is not grounded on semantic roles but syntactic dependencies. Doing so allows us to tackle negation when the negation cue modifies nouns or adjectives.

4 Corpus Creation

We create a corpus of negations and their positive interpretations following three steps. First, we select negations whose negation cue syntactically modifies either a noun or adjective. Second, we automatically generate potential positive from those negations by manipulating syntactic dependencies and part-of-speech tags. Third, we gather manual annotations to validate and score potential interpretations. While asking annotators to suggest positive interpretations would potentially yield more natural interpretations, we found experimentally that a generate-and-rank approach yields higher quality annotations.

Negation in OntoNotes. Instead of building our corpus from plain text, we decided to work on top of OntoNotes (Hovy et al., 2006), a publicly available corpus including texts in several genres (news, transcripts, magazines, etc.).² OntoNotes includes, among other gold linguistic annotations, part-of-speech tags and parse trees. We transformed the parse trees into syntactic dependencies using Stanford CoreNLP (Manning et al., 2014).

We reduce the problem of finding negations to retrieving syntactic dependencies *neg*, which stands for negation modifier. Doing so ignores negation cues that are prefixes or suffixes (e.g., *unlimited*, *motionless*), but also simplifies the process. There are 9,507 *neg* syntactic dependencies in OntoNotes; 6,120 of them modify verbs (64.4%), 1,866 nouns (19.6%), 979 adjectives (10.3%), and 543 other part-of-speech tags (5.7%). Since verbal negation has been tackled

²We use the CoNLL-2011 Shared Task distribution (Pradhan et al., 2011), <http://conll.cemantix.org/2011/>

Noun			
	Negated statement	No condemned murderer has been granted clemency in California since nineteen sixty-seven.	
	Positive counterpart	[A] condemned murderer has been granted clemency in California since nineteen sixty-seven.	
	Relevant tokens	[A] condemned murderer has been granted clemency in California since nineteen sixty-seven.	
	Potential positive interpretations	Intpn. 1, root	[A] condemned murderer has been {some verb} clemency in California since nineteen sixty-seven, but not <i>granted</i> .
		Intpn. 2, nsubjpass	{Someone} has been granted clemency in California since nineteen sixty-seven, but not [<i>a</i>] <i>condemned murderer</i> .
Intpn. 3, dobj		[A] condemned murderer has been granted {something} in California since nineteen sixty-seven, but not <i>clemency</i> .	
Intpn. 4, prep		[A] condemned murderer has been granted clemency {somewhere} since nineteen sixty-seven, but not <i>in California</i> .	
Intpn. 5, prep		[A] condemned murderer has been granted clemency in California {at some point of time}, but not <i>since nineteen sixty-seven</i> .	
Adjective			
	Negated statement	But she was not alive when she got to the Lafayette area.	
	Positive counterpart	But she was alive when she got to the Lafayette area.	
	Relevant tokens	She was alive when she got to the Lafayette area.	
	Potential positive interpretations	Intpn. 1, root	She was {some adjective} when she got to the Lafayette area but not <i>alive</i> .
		Intpn. 2, nsubj	{Somebody} was alive when she got to the Lafayette area but not <i>she</i> .
Intpn. 3, advcl		She was alive {at some point of time} but not <i>when she got to the Lafayette area</i> .	

Table 1: Examples of negations and the steps to generate potential positive interpretations (the negated token is either a noun (top) or adjective (bottom)). We also indicate the dependency between a token in the potential focus and a token outside the potential focus.

before (Section 3), we focus on negation cues that modify nouns or adjectives. We use the term *negated token* to refer to the token that is syntactically modified by the negation cue. The most frequent negated tokens that are nouns or adjectives are plotted in Figure 1.

4.1 Selecting Negations

Annotating all negations that modify a noun or adjective is outside the scope of this paper. To alleviate the annotation effort, we discard negations that belong to sentences that do not have at least one verb and one subject (*nsubj* or *nsubjpass* dependencies), and sentences that contain more than two negations, conditionals, questions or commas. Additionally, we skip negations if the negated token is the noun *one*, as scoring their potential positive interpretations is straightforward. Out of the

1,866 and 979 negation modifying nouns and adjectives, 635 and 320 pass the above filters. Out of these, we randomly select 309 and 75 respectively (approximately 50% and 25%).

4.2 Generating Potential Positive Interpretations

We generate potential positive interpretations automatically using a deterministic procedure that manipulates part-of-speech tags and syntactic dependencies. The first step is to remove the negation cue to obtain the positive counterpart. Then, we use dependencies to select the tokens relevant to the negation—the main motivation is to select the eventuality to which the negation belongs. Finally, we generate potential interpretations using a battery of deterministic rules. Table 1 shows the output of each step with two sample negations.

Selecting Relevant Tokens. Negation may occur in sentences with multiple clauses. We simplify the original sentence and identify the eventuality to which the negation belongs by using the rules below. We defined these rules after analyzing several examples and the Stanford dependencies manual (de Marneffe and Manning, 2008).

When the negated token is a noun, we have two scenarios. Scenario (1) occurs when the negated token is the root or it has a *cop* dependency (*copula*) with the verb *to be*. In this case, we select all the dependents of the negated token.³ Scenario (2) occurs when neither of the two rules above apply. In that case, we select all the dependents of the closest verb, where the closest verb is the first verb found when traversing the dependency tree from the negated token to the root. For example, we simplify *Article three says that no law shall prohibit any religious belief* to *No law shall prohibit any religious belief*.

When the negated token is an adjective, we select all the dependents of the negated token. For example, from *His estimate of 3.3% for third-quarter GNP is higher than the consensus because he believes current inventories aren't as low as official figures indicate*, we select *Current inventories aren't as low as official figures indicate*.

Manipulating Dependencies to Generate Potential Positive Interpretations. After removing the negation cue and selecting relevant tokens, we use syntactic dependencies to select potential foci. Once potential foci are identified, generating positive interpretations is straightforward: each focus yields one interpretation, where everything but the focus is positive. The main idea is to select as potential foci subtrees rooted at selected tokens.

When the negated token is a noun, we select as potential foci the subtrees rooted in all the direct dependents of the negated token in Scenario (1), and the subtrees rooted in all the direct dependents of the closest verb in Scenario (2). When the negated token is an adjective, we select as potential foci the subtrees rooted in all the direct dependents of the negated token. This strategy to select potential foci has a few exceptions to avoid foci that yield meaningless interpretations. Specifically, we discard potential foci:

- whose root has dependency *aux*, *auxpass*, *cop*, *poss*, *dep*, *prt* or *punct*;

³If the negated token is a noun and the root, this scenario is equivalent to selecting the whole sentence

- that consist of
 - the determiner *the*, *a*, *an*, *it* and *there*;
 - the adverbs *so*, *too*, *though*, *even*, *still*, *as*, *quite*, *either*, *however*, *any-more*, *moreover*, *therefore*, *furthermore*, *hence*, *thus*, *further*, *apparently*, *clearly*, *specifically*, *actually*, *fortunately*, and *unfortunately*;
 - a single token with part-of-speech tag TO, CC, UH, POS or IN.

These exceptions were defined after manual examination of several examples. For example, consider sentence *It's not just women and girls who are affected*. We avoid generating interpretations *It's just women and girls who {X} affected* (focus would be *are*, with dependency *aux*), and *It's just women {X} girls who are affected* (focus would be *and*, with part-of-speech tag *CC*).

After potential foci are selected, we generate potential positive interpretations by rewriting each focus with “someone / some people / something / etc.”, and appending “but not text_of_focus” at the end. Table 1 details the steps to generate potential positive interpretations.

4.3 Scoring Potential Interpretations

Once potential interpretations are generated automatically, we manually annotate them. The annotation interface shows the sentence containing the negation, the previous and next sentences as context, and one potential positive interpretation at a time. Annotators are asked *Given the text snippet below [previous sentence, sentence containing the negation and next sentence], do you think the statement [positive interpretation] is true?*, and must answer with a score ranging from 0 to 5, where 0 means *certainly no* and 5 means *certainly yes*. During pilot annotations, we found that certainty must be taken into account as forcing annotators to answer *yes* or *no* proved too restrictive. Note that some negations do not have any positive interpretation scored high, e.g., all interpretations from *Utter no words* receive a low score.

5 Corpus Analysis

Table 2 shows basic counts and statistics of the annotated potential positive interpretations. *Dependency* indicates the syntactic dependency between a token within the potential focus and a token outside the potential focus. The total number of potential positive interpretations is 777 when

	Negation, context (previous and next sentences) and all potential positive interpretations	Score
Noun	Context, previous sentence: Here’s what that judge said. Negation: The victim in this case is not a young child. Context, next sentence: He’s now sixteen years old.	
	- Interpretation 1: The victim in this case is a young {something}, but not <i>a child</i> .	2
	- Interpretation 2: {Something} is a young child, but not <i>The victim in this case</i> .	4
	- Interpretation 3: The victim in this case is a {some adjective} child, but not <i>young</i> .	5
Adjective	Context, previous sentence: She was alive when she left that nursing home. Negation: But she was not alive when she got to the Lafayette area. Context, next sentence: CNN made repeated attempts to contact administrators or representatives of Huntingdon Place.	
	- Interpretation 1: She was {some adjective} when she got to the Lafayette area, but not <i>alive</i> .	5
	- Interpretation 2: {Somebody} was alive when she got to the Lafayette area, but not <i>she</i> .	2
	- Interpretation 3: She was alive {at some point of time}, but not <i>when she got to the Lafayette area</i> .	5

Table 3: Annotation examples. We show the original sentence containing a negation (the negated token is either a noun or an adjective), its context, and all potential interpretations with their scores.

	Dependency	#	%	Score	
				Mean	SD
Nouns	nsubj	231	30.0	4.45	0.88
	root	169	21.7	2.96	1.41
	pobj	70	9.0	3.99	1.31
	amod	50	6.4	4.72	0.8
	ccomp	45	5.7	3.62	1.34
	other	212	27.3	3.73	1.49
	total	777	100.0	3.86	1.38
Adjectives	nsubj	57	28.5	4.12	0.80
	root	50	25.0	4.82	0.48
	ccomp	22	11.0	4.91	0.29
	pobj	21	10.5	4.19	1.05
	xcomp	12	6.0	4.42	1.32
	other	38	19.0	4.08	1.46
total	200	100.0	4.40	0.99	
All		977	100.0	3.97	1.33

Table 2: Corpus analysis. For each dependency, we show the total number and percentage of interpretations, mean score and standard deviation.

the negated token is a noun, and 200 when it is an adjective. On average, we generate 2.5 potential interpretation per negation when the negated token is a noun, and 2.7 when it is an adjective (we selected 309 and 75 negations respectively).

When the negated token is a noun, scores are overall lower (3.86 vs. 4.40), and scores are higher when the *dependency* is either *nsubj* (nominal subject, 4.45) or *amod* (adjectival modifier, 4.72). Regardless of *dependency*, mean scores are always over 4 when the negated token is an adjective.

Annotation Quality. The procedure to generate potential interpretations (Section 4.2) was tuned iteratively until we achieved considerable annotation agreement in pilot annotations. To ensure quality, we calculated Pearson correlations with 20% of annotations, and stopped the refinement process when we achieved 0.76 Pearson correlation. Note that Pearson is better suited than agree-

ment measures designed for categorical labels, as not all disagreements are equally bad, e.g., 4 vs. 1 is worse than 4 vs. 5.

5.1 Annotation Examples

Table 3 presents two complete annotation examples when the negated token is a noun and adjective. We show all potential interpretations generated and the manually assigned scores.

We generate three potential positive interpretations from the first example, *The victim in this case is not a young child*, and two of them received high scores (4 and 5). When reading the statement in context, it is clear that the judge implied that *Something (a younger human) would be a younger child* (Interpretation 2), and that *The victim of this case is an older (not young) child* (Interpretation 3). Interpretation (1), *The victim in this case is a young something*, receives a low score (2 out of 5). One could argue that this interpretations should receive a higher score because *The victim in this case is a young adult*, we simply provide real annotations drawn from our corpus.

The procedure to generate potential positive interpretations also generates three interpretations from the second example, *But she was alive when she got to the Lafayette area*, and two of them receive the highest score (5 out of 5). Interpretation (1) encodes the intuitive meaning that *She was dead when she got to the Lafayette area*, and Interpretation (3) captures that *She was alive before she got to the Lafayette area*. Interpretation (2) receives a low score (2 out of 5), as there is no evidence suggesting which individuals were alive when they got to the Lafayette area.

Type	Name	Description
Basic	neg_cue	word form of the negation cue
	neg_token	word form and part-of-speech tag of negated token
Path	syn_path_dep	path of dependencies from focus to negated token (or verb)
	syn_path_pos	path of POS tags from focus to negated token (or verb)
	last_syn_path_dep	last syntactic dependency in syn_path_dep
	last_syn_path_pos	last part-of-speech tag in syn_path_pos
Focus	focus_length	number of tokens in potential focus
	focus_first_word	word form and part-of-speech tag of first word in focus
	focus_last_word	word form and part-of-speech tag of last word in focus
	focus_direction	flag indicating whether focus occurs before or after neg_token
	focus_head_word	word form of the head of focus
	focus_head_pos	part-of-speech tag of the head of focus
	focus_head_rel	syntactic dependency of the head of focus

Table 4: Features used to assign scores to automatically generated potential interpretations.

Feature set	Gold			Predicted		
	Nouns	Adjectives	All	Nouns	Adjectives	All
neg_cue	-0.10	-0.24	0.06	0.02	-0.26	-0.02
basic	0.12	-0.10	0.11	0.05	-0.39	0.01
basic + path	0.36	0.59	0.40	0.17	0.58	0.24
basic + path + focus	0.36	0.52	0.42	0.33	0.39	0.34

Table 5: Pearson correlations obtained with the test split and several combination of features. We detail results by the part-of-speech tag of the negated token (noun or adjective).

6 Learning to Score Potential Interpretations

We solve the task of scoring potential positive interpretations using standard supervised machine learning. We divide negations and their corresponding interpretations into training (80%) and test (20%), and use SVM with RBF kernel as implemented in scikit-learn (Pedregosa et al., 2011). We tune parameters C and γ using 10-fold cross-validation using the training set.

6.1 Feature Selection

Table 4 lists the full feature set. We extract features from the negated token (noun or adjective), part-of-speech tags and dependency tree.

Basic features are straightforward. They include the negation cue, and the word form and part-of-speech tag of the negated token.

Path features are derived from the syntactic path between the subgraph selected as focus and the negated token or closest verb. If the negated token is a noun, we extract the path between the subgraph and the negated token in Scenario (1), and between the subgraph and the closest verb in Scenario (2) (Section 4.2). If the negated token is an adjective, we extract the path between the subgraph and the negated token. We include two paths (dependencies and part-of-speech tags), and the last dependency and part-of-speech tag.

Focus features characterize the dependency subgraph chosen as focus to generate the potential interpretation. We include the number of tokens, word form and part-of-speech tags of the first and last tokens, and whether the focus occurs before or after the negated token. Additionally, we extract the word form, part-of-speech-tag and dependency of the head of the focus, which we define as the token whose syntactic head is outside the focus.

7 Experimental Results

We perform two kinds of experiments. First, we score all potential positive interpretations automatically generated (Section 7.1). Second, we identify interpretations scored with the highest score, 5 out of 5 (Section 7.2). We always build separate models for nouns and adjectives, and train with gold linguistic information (POS tags and dependencies). We report results on the test set using both gold and predicted linguistic information. For gold, we use the annotations provided with the CoNLL-2011 release, and for auto, we use the output of SyntaxNet (Andor et al., 2016).

7.1 Scoring all Potential Interpretations

We score all potential interpretations using SVM for regression, and calculate Pearson correlation for evaluation purposes. Table 5 shows results obtained with several combinations of features.

When extracting features from gold linguistic

	Gold						Predicted					
	Nouns			Adjectives			Nouns			Adjectives		
	P	R	F	P	R	F	P	R	F	P	R	F
majority baseline	0.00	0.00	0.00	0.47	1.0	0.64	0.00	0.00	0.00	0.31	1.00	0.48
neg_mark	0.66	0.33	0.44	0.47	1.00	0.64	0.00	0.00	0.00	0.31	1.00	0.48
basic	0.66	0.37	0.47	0.47	1.00	0.64	0.62	0.12	0.21	0.31	1.00	0.48
basic + path	0.78	0.64	0.70	0.60	0.95	0.73	0.73	0.60	0.66	0.37	1.00	0.54
basic + path + focus	0.71	0.63	0.67	0.64	0.94	0.76	0.64	0.50	0.56	0.50	0.83	0.62

Table 6: Precision, Recall and F-measure obtained with the test split for instances with the highest score (5 out of 5). Predicting these interpretations correctly allows our models to identify which of the automatically generated potential interpretations are valid given the negation.

information (part-of-speech tags and syntactic dependencies), using only the negation cue as feature or *basic* features (negation cue and negated token) is rather useless (Pearson correlations range from -0.24 to -0.10). Using *basic + path* features yields 0.36 and 0.59 Pearson correlations for nouns and adjectives respectively, and including *focus* features is detrimental (0.36 and 0.52).

When extracting features from predicted linguistic information, we observe a similar trend in Pearson correlations. Results using predicted linguistic information are not directly comparable with those using gold linguistic information. Our methodology to generate potential interpretations relies heavily on syntactic dependencies, and using predicted interpretations implies that some interpretations present in our corpus cannot be automatically generated because of mistakes made by the parser. Out of the 196 potential interpretations in the original test set (20% of 977 annotated interpretations), we evaluate with the 94 interpretation generated with predicted dependencies (48%).

7.2 Identifying Valid Potential Interpretations

While scoring all potential positive interpretation generated is interesting, determining which of those interpretations are certain (scored 5 out of 5) is arguably more useful in a real system. Indeed, an inference tool would ideally extract certain interpretations from negation, and identify other potential interpretations as a byproduct of our generate-and-rank approach.

To estimate performance under this scenario, we approach the task as a standard binary classification task. Interpretations scored 5 receive the positive label, and other interpretations receive the negative label. Table 6 presents results in the test set (Precision, Recall and F-score) for the positive label using the majority baseline and several com-

binations of features, and gold and predicted linguistic information.

The majority baseline fails to detect any interpretation scored with 5 when the negated token is a noun (in this case, the majority of interpretations are not scored 5), and obtains a modest 0.64 F-score when the negated token is an adjective. Using gold linguistic information, *neg_mark* as the only feature or *basic* features improve performance when the negated token is a noun (F-score: 0.44 and 0.47), but does not improve results when the negated token is an adjective. Adding *path* features brings performance up (nouns: 0.70, adjectives: 0.73), and adding *focus* features yields similar results (nouns: 0.67, adjectives: 0.76).

Using predicted linguistic information, we observe the same general trends, but adding *focus* features brings a substantial improvement over *basic + path* for adjectives: 0.54 vs. 0.62.

8 Conclusions

This paper presents a framework to extract positive meaning from negation when the negation cue modifies a noun or adjective. First, we generate potential positive interpretations deterministically. Second, we rank them according to their likelihood. On average, we generate 2.5 potential interpretations when the negated token is a noun, and 2.7 when the negated token is an adjective.

Experimental results show that scoring all potential positive interpretations is challenging, we obtain overall Pearson correlation of 0.42 using features extracted from gold linguistic information (0.36 for nouns and 0.52 for adjectives). But when identifying interpretations annotated with the highest score (5 out of 5), F-scores are relatively high: 0.67 for nouns and 0.76 for adjectives. The latter evaluation is more suitable, as the ultimate goal is to identify valid positive interpretations and discard other potential interpretations generated with our generate-and-rank approach.

References

- Pranav Anand and Craig Martell. 2012. Annotating the focus of negation in terms of questions under discussion. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, ExProM '12, pages 65–69, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.
- Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Ugroningen: Negation detection with discourse representation structures. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 301–309, Montréal, Canada, 7–8 June. Association for Computational Linguistics.
- Simon Blackburn. 2008. *The Oxford Dictionary of Philosophy*. Oxford University Press.
- Eduardo Blanco and Dan Moldovan. 2011. Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 581–589, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Eduardo Blanco and Zahra Sarabi. 2016. Automatic generation and scoring of positive interpretations from negated statements. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1431–1441, San Diego, California, June. Association for Computational Linguistics.
- H. H. Clark and W. G. Chase. 1972. On the process of comparing sentences against pictures. *Cognitive Psychology*, 3(3):472–517, July.
- Isaac Councill, Ryan McDonald, and Leonid Velikovich. 2010. What’s great and what’s not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 51–59, Uppsala, Sweden, July. University of Antwerp.
- Marie-Catherine de Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.
- Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 495–504, Berlin, Germany, August. Association for Computational Linguistics.
- Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL 2010 shared task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the CoNLL2010 Shared Task*, Uppsala, Sweden. Association for Computational Linguistics.
- B. Garner. 2009. *Garner’s Modern American Usage*. Oxford University Press, USA.
- Laurence R. Horn and Heinrich Wansing. 2015. Negation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2015 edition.
- Laurence R. Horn. 1989. *A natural history of negation*. Chicago University Press, Chicago.
- Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA. Association for Computational Linguistics.
- Rodney D. Huddleston and Geoffrey K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, April.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Roser Morante and Walter Daelemans. 2009. A metalearning approach to processing the scope of negation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL '09*, pages 21–29, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Roser Morante and Walter Daelemans. 2012. Conandoyle-neg: Annotation of negation in conandoyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, Istanbul*.
- Roser Morante and Caroline Sporleder. 2012. Modality and negation: An introduction to the special issue. *Comput. Linguist.*, 38(2):223–260, June.
- Ann E. Nordmeyer and Michael C. Frank. 2013. Measuring the comprehension of negation in 2- to 4-year-old children. *Proceedings of the 35th Annual Conference of the Cognitive Science Society, Austin, TX: Cognitive Science Society*.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 99–108, Lisboa, Portugal, September. Association for Computational Linguistics.
- Mats Rooth. 1985. *Association with focus*. Ph.D. thesis.
- Mats Rooth. 1992. A theory of focus interpretation. *Natural language semantics*, 1(1):75–116.
- Zahra Sarabi and Eduardo Blanco. 2016. Understanding negation in positive terms using syntactic dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1108–1118, Austin, Texas, November. Association for Computational Linguistics.
- György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of BioNLP 2008*, pages 38–45, Columbus, Ohio, USA. ACL.
- Ton van der Wouden. 1997. *Negative contexts: collocation, polarity, and multiple negation*. Routledge, London.
- Erik Velldal, Lilja Ovreid, Jonathon Read, and Stephan Open. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Comput. Linguist.*, 38(2):369–410, June.

Metaheuristic Approaches to Lexical Substitution and Simplification

Sallam Abualhaija

Institute of Computer Technology
Hamburg University of Technology
sallam.abualhaija@tu-harburg.de

Tristan Miller and Judith Eckle-Kohler and Iryna Gurevych

Ubiquitous Knowledge Processing Lab (UKP-TUDA/UKP-DIPF/AIPHES)
Department of Computer Science
Technische Universität Darmstadt
<https://www.ukp.tu-darmstadt.de/>

Karl-Heinz Zimmermann

Institute of Computer Technology
Hamburg University of Technology
K.Zimmermann@tu-harburg.de

Abstract

In this paper, we propose using metaheuristics—in particular, simulated annealing and the new D-Bees algorithm—to solve word sense disambiguation as an optimization problem within a knowledge-based lexical substitution system. We are the first to perform such an extrinsic evaluation of metaheuristics, for which we use two standard lexical substitution datasets, one English and one German. We find that D-Bees has robust performance for both languages, and performs better than simulated annealing, though both achieve good results. Moreover, the D-Bees-based lexical substitution system outperforms state-of-the-art systems on several evaluation metrics. We also show that D-Bees achieves competitive performance in lexical simplification, a variant of lexical substitution.

1 Introduction

Lexical substitution is a special case of automatic paraphrasing in which the goal is to provide contextually appropriate replacements for a given word, such that the overall meaning of the context is maintained. The task has applications in question answering, text summarization, sentence compression, information extraction, machine translation, and natural language generation (Androutsopoulos and Malakasiotis, 2010). It is also frequently employed as an *in vivo* evaluation of word sense disambiguation (WSD) systems (McCarthy and

Navigli, 2009; Toral, 2009; Miller et al., 2015), because while lexical substitution requires words to be sense-disambiguated, it does not impose use of a predefined sense inventory.

Past work in WSD, whether or not it forms part of a lexical substitution system, has employed a wide range of approaches (Agirre and Edmonds, 2007). Supervised methods usually achieve the best results, but at the tremendous cost of producing manually annotated training data specific to the language and domain. Knowledge-based and unsupervised methods rely only on pre-existing resources such as machine-readable dictionaries and raw corpora. Though generally less accurate, they have the advantage of being more flexible and more adaptable to new languages and domains. For knowledge-based methods, this has been especially true since the advent of large, multilingual, collaboratively constructed resources such as Wikipedia and Wiktionary (Zesch et al., 2008).

In this paper, we present two novel approaches to lexical substitution which are knowledge-based, generally language-independent, and use a combination of traditional wordnets and Wiktionary. The first approach uses simulated annealing (Kirkpatrick et al., 1983), which was first proposed for use in WSD by Cowie et al. (1992) but has attracted relatively little attention since then. The second approach uses D-Bees (Abualhaija and Zimmermann, 2016), a relatively new, biologically inspired disambiguation algorithm that models swarm intelligence. Both algorithms are *metaheuristic* (Talbi, 2009) in that they treat WSD as an optimization problem and modify heuristic (approximate) solu-

tions to avoid entrapment in local optima. Ours is the first extrinsic evaluation of any metaheuristic approaches to WSD in a lexical substitution setting.

We evaluate and compare both approaches on two lexical substitution datasets, one English and one German. We find that both approaches perform well, with D-Bees in particular exceeding state-of-the-art performance in many tasks. We also apply the systems to lexical simplification, a variant of lexical substitution in which the goal is to provide substitutes which are easier to understand. Here, too, we find that D-Bees performs near or above the state of the art.

2 Background

2.1 Lexical Substitution and Simplification

In lexical substitution, a system is given a word in context and tasked with producing a list of words that could be substituted for the word without altering the overall meaning. For example, given the word “bright” in the sentence “Einstein was a bright man,” valid substitutes would include “sharp” and “intelligent”, but not “shiny” or “luminous”, even though the latter two are synonymous with “bright” in other contexts. It is generally expected that the list of substitutes be ordered by acceptability. Most lexical substitution systems therefore comprise two distinct phases: *generation*, in which the system assembles a set of suitable substitutes for the target word, and *ranking*, in which the system orders them according to how well they fit the context.

There have been a number of organized evaluation campaigns for lexical substitution systems, including the English-language task at SemEval-2007 (McCarthy and Navigli, 2009) and the German task at GermEval 2015 (Miller et al., 2015). These campaigns provide standardized datasets where a large number of word–context combinations have been manually annotated with acceptable substitutes. Systems are evaluated by comparing their output to this gold standard, using any or all of three scoring methodologies:

- In the *best* methodology (McCarthy and Navigli, 2009), systems are allowed to suggest as many substitutes as they wish. However, the credit for each guess is normalized by the total number of guesses. The best guess should be placed first in the list. Across the entire dataset, four metrics are calculated: recall (R), mode recall (R_m), precision (P), and mode

precision (P_m).¹

- In *out of ten (OOT)* (McCarthy and Navigli, 2009), systems suggest up to ten substitutes, though neither the exact number nor the order of these is important. This methodology uses minor variations of *best*’s R , R_m , P , and P_m .
- *Generalized average precision (GAP)* (Kishida, 2005) uses a single metric to score a fully ranked list of substitutes. Unlike *OOT*, *GAP* is sensitive to the relative positions of the correct and incorrect substitutes in the list.

For reasons of space, we do not provide detailed explanations and formulas for the nine metrics, but refer readers to the cited papers.

Lexical simplification is a variant of lexical substitution in which the correct ranking is determined not just by the substitutes’ contextual fitness but also by their simplicity. (For example, rare words are generally considered to be more complex, as readers are less likely to be familiar with their meanings.) As with other types of text simplification, lexical simplification can be used to make complex texts understandable by a wider range of readers, such as children or second language learners.

To date there has been one shared task in lexical simplification (Specia et al., 2012). Its main evaluation metric is based on Cohen’s (1960) κ . Two post-hoc evaluation metrics are also used. The first, *top-ranked (TRnk)*, evaluates the simplest set of substitutes that is ranked first by the system, compared with the top-ranked set of substitutes in the gold standard. This represents the intersection between the first substitute set found by the system with the first set in the gold standard. The intersection should include at least one substitute. The second metric, *recall at n (R@n)* is the ratio of candidates from the top n sets of substitutes to those in the gold standard, where $1 \leq n \leq 3$. For a given n , the contexts with at least $n + 1$ substitutes in the gold standard are considered.

2.2 Word Sense Disambiguation, Optimization, and Metaheuristics

Word sense disambiguation, the task of determining which of a word’s meanings is the one intended in a given context, is a prerequisite for generating substitutes in knowledge-based lexical substitution.

¹These metrics are inspired by, but distinct from, the traditional recall and precision metrics from information retrieval.

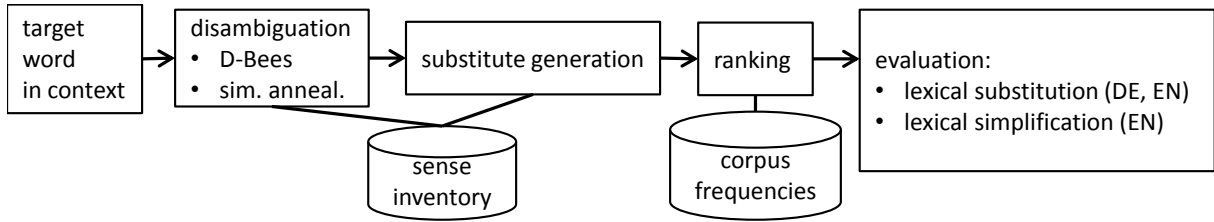


Figure 1: Our approach to lexical substitution and simplification of a target word in context.

There are many different approaches to WSD; for our purposes it is convenient to define it as an optimization problem where the aim is to disambiguate a sequence of words simultaneously (Abualhaija and Zimmermann, 2016): Let $W = (w_1, w_2, \dots, w_n)$ be a sequence of n words to be disambiguated, and $\sigma = (s_1, s_2, \dots, s_n)$ the corresponding sequence of senses for each word. Let $\mathcal{S} = \{\sigma_1, \dots, \sigma_m\}$ be the set of all sequences of senses that represent sense combinations of the words in W . Then the objective function is $\arg \max_{\sigma \in \mathcal{S}} \ell(\sigma)$, where ℓ is the score assigned to a sequence of senses according to some measure of semantic similarity, such as those surveyed by Zesch and Gurevych (2010).

WSD as an optimization problem is NP-hard. This can be worked around by using metaheuristics, which are approximate, tractable algorithms that find near-optimal solutions. Metaheuristics can be *single-solution* and *population-based* search methods. The former manipulate and transform a single solution, giving more focus to the promising regions. Population-based methods work on multiple solutions, distributing their focus and exploring several regions of the search space simultaneously.

3 Approach

We investigate two knowledge-based, language-independent approaches to lexical substitution, whose main difference lies in the metaheuristic WSD component preceding the generation phase. Both approaches use a top-down generation process, in which the target word is first disambiguated in context with respect to a particular sense inventory, and then used to suggest a list of substitutes.² In the following subsections, we describe the two disambiguation components and the common substitute generation and ranking components. (See overview in Figure 1.)

²This contrasts with a bottom-up approach, where a list of all possible substitutes for the target word is first generated and then filtered to suit the context.

3.1 Disambiguation with Simulated Annealing

Simulated annealing (Kirkpatrick et al., 1983) is a single-solution algorithm in which a randomly created solution is iteratively modified until a “good-enough” solution is found. To apply it to WSD, we use essentially the same setup as Cowie et al. (1992). We start with a randomly initialized *sense combination* $\sigma_0 = (s_1, s_2, \dots, s_n)$ from a given sense inventory, for each word in the context. We then retrieve the glosses for each sense, preprocess them via lemmatization and stop word removal, and give each remaining term a score of $n - 1$ if it appears n times. We calculate the configuration’s *redundancy*, R_0 , by summing up all the scores. In other words, R_0 is the lexical overlap between sense definitions. The aim of simulated annealing is to maximize this overlap, or more precisely to minimize the *energy* function $E_i = 1/(1 + R_i)$ in each iteration i .

In this iterative process, each iteration makes a random change on the configuration σ_i to produce σ_{i+1} , on which the corresponding E_i is computed. If $E_{i+1} < E_i$ (i.e., $\Delta E < 0$), then the new configuration replaces the old configuration for the next iteration. Otherwise, the new configuration might still be accepted with probability $\Pr = \exp(-\Delta E/T)$, where T is initially set to 1 but replaced with $0.9T$ for each subsequent iteration. This way, the algorithm risks exploring poor-looking paths that might nonetheless yield better results in the long run, and the earlier the iterations are, the greater the probability that a poor path is followed. In our experiments we iterate up to 30 times.

3.2 Disambiguation with D-Bees

D-Bees (Abualhaija and Zimmermann, 2016) is a population-based algorithm inspired by bee colony optimization (BCO) (Teodorović, 2009). BCO models the foraging behaviour of honey bees, where thousands of individuals with limited knowledge collaborate to maximize their collective bene-

fit. In nature, bees fly around their hive to look for nectar and pollen. When they find it, they return to the hive and perform a dance to advertise its location and quality to the others. The observers then decide whether to remain committed to their own path or to abandon it in favour of one of the advertised paths. BCO simulates this method through a multi-agent decentralized system.

D-Bees starts by choosing one of the target words as the hive, which spawns bee agents and sends them to other words in the context. The number of bee agents equals the number of candidate senses of the hive; each bee agent starts off with one of these senses in its memory. For each word it visits, the bee disambiguates it by randomly selecting a candidate sense, building up a path of senses and maintaining a running total similarity score. This *forward pass* continues until a set number of moves is reached.

The bee then makes a *backward pass* to the hive and exchanges its partial solution with the other agents on the virtual dancing floor. Each bee then determines whether it should stick to its path or adopt that of another bee; this is accomplished through *loyalty* and *recruiting probability* functions that depend mainly on the quality of the partial solutions. On the next forward pass, the bees resume their searches from the ends of their chosen paths. The forward and backward passes are alternated until there are no more words to be disambiguated. The bee agent with the best solution determines the final sense labelling of all words in the context.

In experiments on separate tuning datasets, we determined the number of moves in the forward pass to be one-third the number of context words. For the calculation of semantic similarity, we use a variant of the adapted Lesk algorithm (Banerjee and Pedersen, 2002). For each sense, we build a textual representation by concatenating its gloss with those of its hyper- and hyponyms. We then calculate the lexical overlap between the two texts.

3.3 Substitute Generation

Once the target word is disambiguated with respect to a particular sense inventory, we generate an unordered list of substitutes (to be subsequently ordered by the ranking module). The sense inventory we use for disambiguation is WordNet 3.1 (Fellbaum, 1998) for our English tasks, and GermaNet 10.0 (Hamp and Feldweg, 1997; Henrich and Hinrichs, 2010) for the German one. These are

expert-built resources in which words representing the same concept are grouped together into *synsets*; synsets are in turn linked into a network by semantic relations such as hypernymy and meronymy.

In preliminary experiments on generating substitutes, we varied two independent parameters: which lexical-semantic resources to use as the source of substitutes, and which semantic relations to follow from the disambiguated synset.

With respect to the first parameter, we tried drawing substitutes from the disambiguation inventory (WordNet or GermaNet) alone, and also drawing additional substitutes from Wiktionary. Our use of Wiktionary as a complementary resource is motivated by Meyer and Gurevych (2012), who found its coverage to be complementary to those of expert-built resources, and by Henrich and Hinrichs (2012), who found that using information from both GermaNet and Wiktionary improved WSD performance. We used a relatively simple, Lesk-like method for mapping senses from WordNet/GermaNet to Wiktionary.

For the second parameter, we tried one setup in which we took all synonyms found in the disambiguated synset and in its hypernyms, and one in which we additionally pulled in synonyms from the hyponyms and all other related synsets (except antonyms). The first setup was informed by the annotation guidelines of the lexical substitution datasets, which indicate that it is permissible to suggest substitute terms that are more generic but not more specific. The second setup was informed by the analyses of Kremer et al. (2014) and Miller et al. (2016), which found, contrarily, that other semantic relations, including hyponyms, were a fruitful source of substitutes.

We obtained the best overall results when using both WordNet/GermaNet and Wiktionary, and when following semantic relations of all types (other than antonymy), to build the substitute list. We therefore used this setup for all our lexical substitution and simplification experiments.

3.4 Ranking

The final step of lexical substitution is to rank the substitutes. Our method, like those employed in previous lexical substitution tasks, assumes that a substitute's suitability depends on the type of its semantic relation to the target word. We therefore order the substitutes as follows: synonyms, hypernyms, hyponyms, other relations. Within

each semantic relation type, we sort the substitutes first by source (first WordNet/GermaNet, then Wiktionary), and then secondarily by reverse frequency in a large corpus. In preliminary experiments, we found that this method was generally better than simply sorting the entire substitute list by reverse frequency. To determine lemma frequency, we use the same frequency lists used to construct the original datasets: WaCky (Baroni et al., 2009) for German, and BNC (Burnard, 2007) for English.

4 Lexical Substitution for German

4.1 Dataset and Baselines

In our experiments, we use the data from GermEval 2015 (Miller et al., 2015), a shared task for German-language lexical substitution. It is split into a training and a test set of 1040 and 1000 sentences from the German edition of Wikipedia. Each sentence in the dataset contains one of 75 unique target words (25 nouns, 25 verbs, and 25 adjectives); in the test set, ten sentences are provided for each of the nouns and adjectives, and twenty for each verb.

Miller et al. (2015) report results of several naïve baselines, the best-performing of which are *weighted sense* (Toral, 2009) and *top-ranked synonym* (McCarthy and Navigli, 2009). Neither baseline makes any attempt to disambiguate the target word; rather, they build a substitute list by gathering synonyms of all possible senses of the target, as well as synonyms of closely related senses such as hypernyms, and then ranking these words by their frequency (either within the list itself or in a large corpus). We consider these two naïve baselines as reasonable lower bounds.

The more challenging baseline performance comes from the best-performing participating systems at GermEval 2015, which represent the state of the art in German-language lexical substitution. One of these systems (Hintz and Biemann, 2015) is a supervised, bottom-up approach inspired by previous English-language work by Szarvas et al. (2013a). It first retrieves a list of substitutes from various lexicons, then applies a maxent classifier to determine whether each substitute fits the context. The second system (Jackov, 2015) is based on techniques from machine translation. It first disambiguates the input text by mapping German words to concepts represented by WordNet synsets. It then produces and scores various parsing hypotheses, and selects the synonyms and hypernyms of

the target in the best-scoring hypothesis.

4.2 Results and Analysis

Table 1 shows the results of the baselines described above, along with those of our basic D-Bees- and simulated annealing-based systems, and an enhanced version of the D-Bees system that we describe below.³ Both our basic systems outperform the prior state of the art for the four *OOT* metrics, with the D-Bees-based system performing slightly better than the one using simulated annealing. However, neither system was able to beat Hintz and Biemann (2015) for the *GAP* and *best* metrics.

In light of this gap, we modified the D-Bees-based system to account for some idiosyncrasies of our German-language resources:

- Where GermaNet provided additional spellings of a synonym (e.g., “*wacklig*” for “*wackelig*”), we placed the variant spellings at the end of the substitute list. This prevented the top ranks of the list from being overloaded with nearly identical terms.
- Where our resources provided gender-specific variants of a synonym, we filtered out those that did not match the gender of the target. For example, when building the substitute list for “*Meisterin*” (female champion), we exclude “*Meister*” (male champion), even though GermaNet lists it as a synonym.
- To control for Wiktionary’s lack of consistency, we filtered out Wiktionary-derived synonyms where the synonymy relation was not symmetric. For example, the Wiktionary entry for “*Likör*” gives “*Crème*” as a synonym, but the entry for “*Crème*” does not give “*Likör*”, so when building a substitute list for “*Likör*”, we do not include “*Crème*”.

With these resource-specific enhancements, the D-Bees system achieves state-of-the-art performance not only for *OOT* but also for *GAP*, and performs only slightly worse than Hintz and Biemann (2015) for *best*. (This is an impressive result considering that Hintz and Biemann (2015) is a supervised system while ours is based solely on external knowledge bases and does not require any training data.) We also examined its performance by part of speech. We found that it remains the

³Here, as well as in §5, we report results on the test split and used the training split for tuning our algorithms.

Table 1: System performance on the GermEval 2015 lexical substitution dataset.

System	Best				OOT				GAP
	<i>P</i>	<i>R</i>	<i>P_m</i>	<i>R_m</i>	<i>P</i>	<i>R</i>	<i>P_m</i>	<i>R_m</i>	
D-Bees	7.66	7.66	14.85	14.85	20.68	20.68	37.73	37.73	12.94
D-Bees (enhanced)	10.39	10.39	22.39	22.39	21.88	21.88	39.64	39.64	16.40
simulated annealing	9.40	9.40	19.67	19.67	19.95	19.95	36.16	36.16	14.34
Hintz and Biemann (2015)	11.20	11.10	24.28	24.21	19.49	19.31	33.99	33.89	15.96
Jackov (2015)	6.73	6.45	13.36	12.86	20.14	19.32	33.18	31.92	11.26
top-ranked synonyms	10.04	10.04	19.82	19.82	15.21	15.21	27.99	27.99	12.25
weighted sense	7.50	7.50	13.46	13.46	20.54	20.54	35.55	35.55	14.28

best-performing system for *GAP* across all parts of speech, and for nouns and verbs is able to match or exceed Hintz and Biemann (2015) on some *best* metrics.

5 Lexical Substitution for English

5.1 Dataset and Baselines

Our English-language data is taken from the SemEval-2007 shared task (McCarthy and Navigli, 2009). That task uses a sample of 201 target words (nouns, verbs, adjectives and adverbs); for each word, ten context sentences are selected from the English Internet Corpus (Sharoff, 2006). Five human annotators provided up to three substitutes for each target. The dataset is split into a training set (300 sentences) and a test set (1710 sentences).

McCarthy and Navigli (2009) provide results for the aforementioned “top-ranked synonyms” algorithm as a lower bound on performance. State-of-the-art performance across the nine evaluation metrics is represented by the top-performing systems at SemEval-2007 (Giuliano et al., 2007; Hassan et al., 2007; Yuret, 2007; Zhao et al., 2007) and by several later systems (Biemann and Riedl, 2013; Melamud et al., 2015).⁴ Of these systems, only Yuret (2007) is supervised.

5.2 Results and Analysis

Table 2 shows the results for the state-of-the-art and naïve baselines, along with results of our two basic systems and, as before, an enhanced version

⁴We are aware of several further lexical substitution systems (Moon and Erk (2013), Ó Séaghdha and Korhonen (2014), Roller and Erk (2016), Sinha and Mihalcea (2011), Szarvas et al. (2013b), and Thater et al. (2010) as reimplemented by Kremer et al. (2014)), though they do not report results on the full SemEval-2007 test set, or else do not report any of the same metrics we do, or else are concerned only with ranking but not generating substitutes.

of the D-Bees system. Our systems’ performance is generally much lower here than on the German-language data, with D-Bees failing to exceed the state of the art.

As with our German experiments, we tried modifying the D-Bees-based system to work around the language-specific problems we observed. The most significant of these adaptations are as follows:

- Our analysis suggested that WordNet’s notoriously fine sense granularity was adversely affecting the WSD process. We therefore modified D-Bees to perform “soft” WSD (Ramakrishnan et al., 2004), meaning that we allow it to select several different senses as the correct ones—in our case, up to five. To compensate for the larger number of substitution candidates, we limit the ranked list of substitutes to 20. (This harkens back to the bottom-up approaches defined in §3.) Substitutes generated from the best disambiguation solution are ranked highest.
- In contrast to German, English lexical substitutes are often drawn from indirect hypernyms (Kremer et al., 2014; Miller et al., 2016). (This too may be an artifact of WordNet’s fine granularity.) We therefore extended our substitute search to two levels of hypernyms.
- The glosses provided by WordNet sometimes consist of a list of equivalent terms which do not appear in the list of synonyms. For example, WordNet defines one sense of the adverb “right” as “precisely, exactly”, though it does not actually list those words as synonyms. We therefore include as the lowest-ranked substitutes those words from the target’s gloss that match its part of speech.

Table 2: System performance on the SemEval-2007 lexical substitution dataset.

System	Best				OOT				GAP
	P	R	P_m	R_m	P	R	P_m	R_m	
D-Bees	8.73	8.73	14.88	14.88	24.88	24.88	35.53	35.53	13.25
D-Bees (enhanced) simulated annealing	11.77	11.77	19.35	19.35	34.68	34.68	47.80	47.80	17.93
Zhao et al. (2007)	11.35	11.35	18.86	18.86	33.88	33.88	46.91	46.91	—
Giuliano et al. (2007)	6.95	6.94	20.33	20.33	69.03	68.90	58.54	58.54	—
Yuret (2007)	12.90	12.90	20.65	20.65	46.15	46.15	61.30	61.30	—
Hassan et al. (2007)	12.77	12.77	20.73	20.73	49.19	49.19	66.26	66.26	—
Melamud et al. (2015)	8.09	8.09	13.41	13.41	27.65	27.65	39.19	39.19	—
Biemann and Riedl (2013)	—	—	—	—	27.48	27.48	37.19	37.19	—
top-ranked synonyms	9.95	9.95	15.28	15.28	29.70	29.35	40.57	40.57	—

- As WordNet contains no hypernymy relations for adjectives, for our purposes we use its “similar-to” relation instead.
- For word frequency, we generally prefer the counts provided by WordNet, since they are sense-disambiguated. (This use of manually sense-annotated data makes our approach weakly supervised.) In other cases, such as when ranking substitutes from Wiktionary, we use Web 1T (Brants and Franz, 2006) instead of BNC. Web 1T is a much larger, more modern, Web-derived corpus that may better reflect the lemma distributions in the Web-derived SemEval-2007 dataset.

The enhanced D-Bees-based system performs significantly better than the base system, though in common with the two post-SemEval-2007 systems, it still fails to surpass the state of the art for *best* and *OOT*. The two knowledge-based systems that outperform our system by a large margin, Giuliano et al. (2007) and Hassan et al. (2007), employ particularly strong substitute generation components that use a combination of WordNet with a rich thesaurus resource—the *Oxford American Writer Thesaurus* and the *Microsoft Encarta* encyclopedia, respectively. Both resources outperform Wiktionary in terms of coverage of synonyms and semantically related words. However, as these resources are proprietary, they were not available to us.

Our system’s performance is roughly on par with Zhao et al. (2007), another bottom-up approach. Our enhanced system does achieve the highest known *GAP* score, though this is largely because most prior work does not use this metric, or else

applies it only to the ranking of gold-standard substitutes.

6 Lexical Simplification

6.1 Experimental Setup

Our experiments use the dataset from the SemEval-2012 English lexical simplification task (Specia et al., 2012). It uses the same contexts and target words as the SemEval-2007 dataset, but the gold-standard substitutes, which include the original target words, have been manually re-ranked according to their perceived simplicity. Unlike SemEval-2007, the SemEval-2012 task is concerned exclusively with ranking substitutes; all the original participating systems were given the gold-standard substitutes and simply asked to put them in the correct order. However, to score our own systems we use their own substitute lists, removing only those substitutes that do not also appear in the gold-standard list. This puts us at somewhat of a disadvantage, since our substitute lists often contain only a subset of the gold-standard substitutes. It also makes use of the κ metric problematic, since κ expects the system and gold-standard lists to contain the same set of substitutes. We therefore report only *TRnk* and *R@n* scores.

Specia et al. (2012) report scores for two lower-bound baselines: one puts the substitute lists in random order, and the other orders them by inverse frequency of occurrence in Web 1T.⁵ The state of the art is represented by Jauhar and Specia (2012),

⁵A third baseline leaves the lists in their original order (i.e., by inverse number of annotators who chose them). We ignore it here as it relies entirely on manual labelling.

Table 3: System performance on the SemEval-2012 lexical simplification dataset.

System	TRnk	R@1	R@2	R@3
D-Bees (enhanced) (original ordering)	37.5	71.6	75.5	76.4
D-Bees (enhanced) (unigram ordering)	50.9	72.8	75.2	76.3
D-Bees (enhanced) (n -gram ordering)	47.1	71.3	74.5	75.7
Jauhar and Specia (2012)	60.2	57.5	68.9	76.9
unigram ordering baseline	58.5	55.9	68.1	76.0
random ordering baseline	34.0	32.1	61.2	82.5

a supervised system that classifies substitutes using a context-sensitive n -gram frequency model, a bag-of-words model, and psycholinguistic features. At SemEval-2012 it achieved the best performance for every metric except $R@3$, where it was beaten only by the random baseline.

We first calculated the proportion of instances for which our systems suggested at least one substitute appearing in the gold standard (other than the target word itself). For the simulated annealing system, the percentage was 45.7%, for the D-Bees system it was 58.7%, and for the enhanced D-Bees system, it was 81.6%. We tentatively conclude that the soft WSD of enhanced D-Bees is necessary to generate sufficient numbers of substitutes in common with the gold standard, and exclude our other two systems from further consideration.

Since the SemEval-2012 lexical simplification task is concerned only with ranking, we test three different rankings of the enhanced D-Bees substitute list. First, we preserve the original order of the system. Second, we order by unigram frequency in Web 1T, as in the SemEval-2012 baseline. Our third ranking is an n -gram ordering approach that we found to work well ($\kappa = 0.461$) on the full gold-standard substitute lists. Here the substitutes are sorted according to the summation of the combined frequency of the substitute and context words. More formally, let W be the set of all unique words in the context window, excluding the target w_t , and let S be the set of substitutes for w_t . Then each substitute $s \in S$ is given a score

$$F(s) = \sum_{w \in W} f(s, w),$$

where $f(s, w)$ is the Web 1T co-occurrence frequency for s and w . The list of substitutes is then sorted by descending score.

6.2 Results and Analysis

Table 3 shows the published results for our baselines, along with the results from the enhanced D-Bees-based system from §5.2 using various ranking methods. While none of our configurations scored particularly well on $TRnk$, all of them surpassed the state of the art for $R@1$ and $R@2$, and performed about as well as Jauhar and Specia (2012) for $R@3$. These results are particularly impressive in light of the fact that the SemEval-2012 systems had access to the gold-standard substitutes, whereas our systems did not.

The good $R@n$ scores when using the original ordering indicate that the D-Bees-based system is (quite serendipitously) predisposed to selecting simple substitutes and ranking them relatively highly. We note that there is relatively little difference between our three system configurations, suggesting that all three ranking methods are doing more or less the same thing, at least for the first few substitutes. This result is somewhat surprising in light of Specia et al.’s (2012) assumption that the notion of simplicity is context-dependent. (It is this notion that our n -gram-based ranking model was attempting to capture.) It could be that, for our systems, the context (including text complexity) is already sufficiently accounted for during WSD.

7 Conclusion

In this paper, we have presented the first extrinsic evaluations of simulated annealing and D-Bees in a lexical substitution setting. We used each algorithm as the WSD component in the same knowledge-based, language-independent lexical substitution system. The systems were tested on German and English datasets, and surpassed state-of-the-art performance on the former. The D-Bees system generally had better results, so we applied some resource-specific adaptations based on our own observations of GermaNet and WordNet, as well as on previ-

ously published studies on German and English lexical substitution. These adaptations led to dramatic improvements in performance on both datasets. We also tested the adapted D-Bees system in a lexical simplification setting, where (in spite of some handicaps) it exceeded state-of-the-art performance on two evaluation metrics. Our findings would seem to validate the utility of metaheuristic approaches for lexical substitution and simplification, with the caveat that optimal performance is achieved only when the systems are adapted to the language or linguistic resources used. This adaptation effort may nonetheless be lower than that required to source annotated training data for supervised approaches.

Regarding future work, there are several issues of interest. The first concerns our use of collaboratively constructed language resources. While our WSD components used only expert-built resources, we found it beneficial to draw additional substitution candidates from Wiktionary. For this we used a very basic sense alignment technique, though a more profound sense mapping between WordNet/GermaNet and Wiktionary, such as those surveyed by Gurevych et al. (2016), might lead to better downstream results. The approach D-Bees uses for calculating sense similarity is also quite basic; though it seemed to work well in practice, we are keen to investigate other methods, such as taking the WordNet/GermaNet graph structure into account, or using other measures of text similarity to compare glosses.

Acknowledgments

This work was supported in part by the research training group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES, GRK 1994/1) and by the German Institute for Educational Research (DIPF).

References

- Sallam Abualhaija and Karl-Heinz Zimmermann. 2016. D-Bees: A novel method inspired by bee colony optimization for solving word sense disambiguation. *Swarm and Evolutionary Computation*, 27:188–195.
- Eneko Agirre and Philip Edmonds. 2007. *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech, and Language Technology*. Springer.
- Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entail-

ment methods. *Journal of Artificial Intelligence Research*, 38(1):135–187, May.

- Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: Third International Conference, CIC-Ling 2002*, volume 2276 of *Lecture Notes in Computer Science*, pages 136–145. Springer.

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

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! A framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95, April.

Thorsten Brants and Alex Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium.

Lou Burnard, 2007. *Reference Guide for the British National Corpus (XML Edition)*. British National Corpus Consortium, February.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April.

Jim Cowie, Joe Guthrie, and Louise Guthrie. 1992. Lexical disambiguation using simulated annealing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 1992)*, volume 1, pages 359–365.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. 2007. FBK-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 145–148, June.

Iryna Gurevych, Judith Eckle-Kohler, and Michael Matuschek, 2016. *Linked Lexical Knowledge Bases: Foundations and Applications*, volume 34 of *Synthesis Lectures on Human Language Technologies*, chapter 3: Linking Algorithms, pages 29–44. Morgan & Claypool.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet – A lexical-semantic net for German. In *Proceedings of the ACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. UNT: SubFinder: Combining knowledge sources for automatic lexical

- substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 410–413, June.
- Verena Henrich and Erhard Hinrichs. 2010. GernEdiT – The GermaNet editing tool. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 2228–2235.
- Verena Henrich and Erhard Hinrichs. 2012. A comparative evaluation of word sense disambiguation algorithms for German. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 576–583.
- Gerold Hintz and Chris Biemann. 2015. Delexicalized supervised German lexical substitution. In *Proceedings of GermEval 2015: LexSub*, pages 11–16.
- Luchezar Jackov. 2015. Lexical substitution using deep syntactic and semantic analysis. In *Proceedings of GermEval 2015: LexSub*, pages 17–20.
- Sujay Kumar Jauhar and Lucia Specia. 2012. UOW-SHEF: SimpLex – Lexical simplicity ranking based on contextual and psycholinguistic features. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantic (SemEval 2012)*, volume 2, pages 477–481, June.
- Scott Kirkpatrick, Charles D. Gelatt, Jr., and Mario P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680, May.
- Kazuaki Kishida. 2005. Property of average precision and its generalization: an examination of evaluation indicator for information retrieval experiments. Technical Report NII-2005-014E, National Institute of Informatics, Tokyo, October.
- Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us – Analysis of an “all-words” lexical substitution corpus. In *14th Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference (EACL 2014)*, pages 540–549, April.
- Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159, June.
- Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, June.
- Christian M. Meyer and Iryna Gurevych. 2012. Wiktionary: A new rival for expert-built lexicons? Exploring the possibilities of collaborative lexicography. In Sylviane Granger and Magali Paquot, editors, *Electronic Lexicography*, pages 259–291. Oxford University Press.
- Tristan Miller, Darina Benikova, and Sallam Abualhaija. 2015. GermEval 2015: LexSub – A shared task for German-language lexical substitution. In *Proceedings of GermEval 2015: LexSub*, pages 1–9.
- Tristan Miller, Mohamed Khemakhem, Richard Eckart de Castilho, and Iryna Gurevych. 2016. Sense-annotating a lexical substitution data set with Ubyline. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 828–835, May.
- Taesun Moon and Katrin Erk. 2013. An inference-based model of word meaning in context as a paraphrase distribution. *ACM Transactions on Intelligent Systems and Technology*, 4(3):42:1–42:28, June.
- Diarmuid Ó Séaghdha and Anna Korhonen. 2014. Probabilistic distributional semantics with latent variable models. *Computational Linguistics*, 40(3):587–631, September.
- Ganesh Ramakrishnan, B. P. Prithviraj, A. Deepa, Pushpak Bhattacharyya, and Soumen Chakrabarti. 2004. Soft word sense disambiguation. In *Proceedings of the 2nd Global WordNet Conference (GWC 2004)*, pages 291–298.
- Stephen Roller and Katrin Erk. 2016. PIC a different word: A simple model for lexical substitution in context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1121–1126, June.
- Serge Sharoff. 2006. Open-source corpora: Using the Net to fish for linguistic data. *International Journal of Corpus Linguistics*, 11(4):435–462.
- Ravi Sinha and Rada Mihalcea. 2011. Explorations in lexical sample and all-words lexical substitution. *Natural Language Engineering*, 1(1):1–27.
- Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval-2012)*, pages 347–355.
- György Szarvas, Chris Biemann, and Iryna Gurevych. 2013a. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 10th Conference of the North American Chapter of the Association for Computational Linguistics and the 18th Conference on Human Language Technologies (NAACL-HLT 2013)*, pages 1131–1141.
- György Szarvas, Róbert Busa-Fekete, and Eyke Hüllermeier. 2013b. Learning to rank lexical substitutions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1926–1932, October.
- El-Ghazali Talbi. 2009. *Metaheuristics: From Design to Implementation*. Wiley.

- Dušan Teodorović. 2009. Bee colony optimization (BCO). In Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri, editors, *Innovations in Swarm Intelligence*, pages 39–60. Springer.
- Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 948–957.
- Antonio Toral. 2009. The lexical substitution task at EVALITA 2009. In *Proceedings of the 11th Conference of the Italian Association for Artificial Intelligence*.
- Deniz Yuret. 2007. KU: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 207–214, June.
- Torsten Zesch and Iryna Gurevych. 2010. Wisdom of crowds versus wisdom of linguists – measuring the semantic relatedness of words. *Natural Language Engineering*, 16(1):25–59, January.
- Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, volume 8, pages 1646–1652.
- Shiqi Zhao, Lin Zhao, Yu Zhang, Ting Liu, and Sheng Li. 2007. HIT: Web based scoring method for English lexical substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 173–176, June.

Paraphrasing Revisited with Neural Machine Translation

Jonathan Mallinson, Rico Sennrich and Mirella Lapata

Institute for Language, Cognition and Computation

School of Informatics, University of Edinburgh

10 Crichton Street, Edinburgh EH8 9AB

J.Mallinson@ed.ac.uk, {rsennric,mlap}@inf.ed.ac.uk

Abstract

Recognizing and generating paraphrases is an important component in many natural language processing applications. A well-established technique for automatically extracting paraphrases leverages bilingual corpora to find meaning-equivalent phrases in a single language by “pivoting” over a shared translation in another language. In this paper we revisit bilingual pivoting in the context of neural machine translation and present a paraphrasing model based purely on neural networks. Our model represents paraphrases in a continuous space, estimates the degree of semantic relatedness between text segments of arbitrary length, or generates candidate paraphrases for any source input. Experimental results across tasks and datasets show that neural paraphrases outperform those obtained with conventional phrase-based pivoting approaches.

1 Introduction

Paraphrasing can be broadly described as the task of using an alternative surface form to express the same semantic content (Madnani and Dorr, 2010). Much of the appeal of paraphrasing stems from its potential application to a wider range of NLP problems. Examples include query and pattern expansion (Riezler et al., 2007), summarization (Barzilay, 2003), question answering (Lin and Pantel, 2001), semantic parsing (Berant and Liang, 2014), semantic role labeling (Woodsend and Lapata, 2014), and machine translation (Callison-Burch et al., 2006).

Most of the recent literature has focused on the automatic extraction of paraphrases from various different types of corpora consisting of parallel, non-parallel, and comparable texts. One of the most successful proposals uses bilingual parallel corpora to induce paraphrases based on techniques from phrase-based statistical machine translation (SMT, Koehn et al. (2003)). The intuition behind

Bannard and Callison-Burch’s (2005) bilingual pivoting method is that two English strings e_1 and e_2 that translate to the same foreign string f can be assumed to have the same meaning. The method then pivots over f to extract $\langle e_1, e_2 \rangle$ as a pair of paraphrases. Drawing inspiration from syntax-based SMT, several subsequent efforts (Callison-Burch, 2008; Ganitkevitch et al., 2011) extended this technique to syntactic paraphrases leading to the creation of PPDB (Ganitkevitch et al., 2013; Ganitkevitch and Callison-Burch, 2014), a large-scale paraphrase database containing over a billion of paraphrase pairs in 23 different languages.

In this paper we revisit the bilingual pivoting approach from the perspective of neural machine translation, a new approach to machine translation based purely on neural networks (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Sutskever et al., 2014; Luong et al., 2015). At its core, NMT uses a deep neural network trained end-to-end to maximize the conditional probability of a correct translation given a source sentence, using a bilingual corpus. NMT models have obtained state-of-the-art performance for several language pairs (Jean et al., 2015b; Luong et al., 2015), using only parallel data for training, and minimal linguistic information. In this paper we show how the bilingual pivoting method can be ported to NMT and argue that it offers at least three advantages over conventional methods. Firstly, our neural paraphrasing model learns continuous space representations for phrases and sentences (aka *embeddings*) that can be usefully incorporated in downstream tasks such as recognizing textual similarity and entailment. Secondly, the proposed model is able to either score a pair of paraphrase candidates (of arbitrary length) and generate target paraphrases for a given source input. Due to the architecture of NMT, generation takes advantage of wider context compared to phrase-based approaches: target paraphrases are predicted based on the meaning of the source input and all previously generated target words.

In the remainder of the paper, we introduce our

paraphrase model and experimentally compare it to the phrase-based pivoting approach. We evaluate the model’s paraphrasing capability both *intrinsically* in a paraphrase detection task (i.e., decide the degree of semantic similarity between two sentences) and *extrinsically* in a generation task. Across tasks and datasets our results show that neural paraphrases yield superior performance when assessed automatically and by humans.

2 Related Work

The literature on paraphrasing is vast with methods varying according to the type of paraphrase being induced (lexical or structural), the type of data used (e.g., monolingual or parallel corpus), the underlying representation (surface form or syntax trees), and the acquisition method itself. For an overview of these issues we refer the interested reader to Madnani and Dorr (2010). We focus on bilingual pivoting methods and aspects of neural machine translation pertaining to our model. We also discuss related work on paraphrastic embeddings.

Bilingual Pivoting Paraphrase extraction using bilingual parallel corpora was proposed by Barnard and Callison-Burch (2005). Their method first extracts a bilingual phrase table and then obtains English paraphrases by pivoting through foreign language phrases. Paraphrases for a given phrase are ranked using a paraphrase probability defined in terms of the translation model probabilities $P(f|e)$ and $P(e|f)$ where f and e are the foreign and English strings, respectively.

Motivated by the wish to model sentential paraphrases, follow-up work focused on syntax-driven techniques again within the bilingual pivoting framework. Extensions include representing paraphrases via rules obtained from a synchronous context free grammar (Ganitkevitch et al., 2011; Madnani et al., 2007) as well as labeling paraphrases with linguistic annotations such as CCG categories (Callison-Burch, 2008) and part-of-speech tags (Zhao et al., 2008).

In contrast, our model is syntax-agnostic, paraphrases are represented on the surface level without knowledge of any underlying grammar. We capture paraphrases at varying levels of granularity, words, phrases or sentences without having to *explicitly* create a phrase table.

Neural Machine Translation There has been a surge of interest recently in repurposing sequence transduction neural network models for machine

translation (Sutskever et al., 2014). Central to this approach is an encoder-decoder architecture implemented by recurrent neural networks. The encoder reads the source sequence into a list of continuous-space representations from which the decoder generates the target sequence. An attention mechanism (Bahdanau et al., 2014) is used to generate the region of focus during decoding.

We employ NMT as the backbone of our paraphrasing model. In its simplest form our model exploits a one-to-one NMT architecture: the source English sentence is translated into k candidate foreign sentences and then back-translated into English. Inspired by multi-way machine translation which has shown performance gains over single-pair models (Zoph and Knight, 2016; Dong et al., 2015; Firat et al., 2016a), we also explore an alternative pivoting technique which uses multiple languages rather than a single one. Our model inherits advantages from NMT such as a small memory footprint and conceptually easy decoding (implemented as beam search). Beyond paraphrase generation, we experimentally show that the representations learned by our model are useful in semantic relatedness tasks.

Paraphrastic Embeddings The successful use of word embeddings in various NLP tasks has provided further impetus to use paraphrases. Wieting et al. (2015) take the paraphrases contained in PPDB and embed them into a low-dimensional space using a recursive neural network similar to Socher et al. (2013). In follow-up work (Wieting et al., 2016), they learn sentence embeddings based on supervision provided by PPDB. In our approach, embeddings are learned as part of the model and are available for any-length segments making use of no additional machinery beyond NMT itself.

3 Neural Paraphrasing

In this section we present PARANET, our **Paraphrasing** model based on **Neural Machine Translation**. PARANET uses neural machine translation to first translate from English to a foreign pivot, which is then back-translated to English, producing a paraphrase. In the following, we briefly overview the basic encoder-decoder NMT framework and then discuss how it can be extended to paraphrasing.

3.1 NMT Background

In the neural encoder-decoder framework for MT (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015), the encoder, a recurrent neural network (RNN), is used to compress the meaning of the source sentence into a sequence of vectors. The decoder, a conditional RNN language model, generates a target sentence word-by-word. For the language pair, an encoder takes in a source sentence $X = \{x_1, \dots, x_{T_X}\}$, as a sequence of linguistic symbols and produces a sequence of context vectors $C = \{h_1, \dots, h_{T_X}\}$. PARANET uses a bi-directional RNN, where each context vector h_t is the concatenation of the forward and the backward RNN’s hidden states at time t .

The decoder is a conditional RNN language model that produces, given the source sentence, a probability distribution over the translation. At each time step t' , the decoder’s hidden state is updated:

$$z_{t'} = \text{RNN}(z_{t'-1}, y_{t'-1}, c_{t'}) \quad (1)$$

The update uses the previous hidden state $z_{t'-1}$, the previous target symbol $y_{t'-1}$ and the time dependent context $c_{t'}$, which is computed by an attention mechanism $\alpha_{t',t}$ over the source sentences’ context vectors:

$$c_{t'} = \sum_{t=1}^{T_X} \alpha_{t',t} h_t \quad (2)$$

$$\alpha_{t',t} \propto e^{f(z_{t'-1}, h_t)} \quad (3)$$

g is a feedforward neural network with a softmax activation function in the output layer which returns the probability of the next target symbol. The probability of the target sentence $Y = \{y_1, \dots, y_{T_Y}\}$, is the product of the probabilities of the symbols within the sentence:

$$P(Y|X) = \prod_{t'=1}^{T_Y} P(y_{t'} | y_{<t'}, X) \quad (4)$$

3.2 Pivoting

Pivoting is often used in machine translation to overcome the shortage of parallel data, i.e., when there is not a translation path from the source language to the target. Instead, pivoting takes advantage of paths through an intermediate language. The idea dates back at least to Kay (1997), who observed that ambiguities in translating from one language onto another may be resolved if a translation into some third language is available, and has met with success in traditional phrase-based SMT (Wu and Wang, 2007; Utiyama and Isahara, 2007)

and more recently in neural MT systems (Firat et al., 2016b).

In the case of paraphrasing, there is not a path from English to English. Instead, a path from English to French to English can be used. In other words, we translate a source sentence into a pivot language and then translate the pivot back into the source language. Pivoting using NMT ensures that the entire sentence is considered when choosing a pivot. The fact that contextual information is considered when translating, allows for a more accurate pivoted sentence. It also places greater emphasis on capturing the meaning of the sentence, which is a key part of paraphrasing.

A naive approach to pivoting is one-to-one back-translation. The source English sentence E_1 , is translated into a single French sentence F . Next, F is translated back into English, giving a probability distribution over English sentences, E_2 . This translation distribution acts as the paraphrase distribution $P(E_2|E_1, F)$:

$$P(E_2|E_1, F) = P(E_2|F) \quad (5)$$

One-to-one back-translating offers an easy way to paraphrase, because existing NMT systems can be used with no additional training or changes. However, there are several disadvantages; for example the French sentence F must fully capture the exact meaning of E_1 , as E_1 and E_2 are conditionally independent given F . Since there is rarely a clear one-to-one mapping between sentences in different languages, information about the source sentence can be lost, leading to inaccuracies in the paraphrase probabilities. To avoid this, we propose back-translating through multiple sentences within one and multiple foreign languages.

Multi-pivoting PARANET pivots through the set of K -best translations $\mathcal{F} = \{F_1, \dots, F_K\}$ of E_1 . This ensures that multiple aspects (semantic and syntactic) of the source sentence are captured. Moreover, multiple pivots provide resilience against a single bad translation, which would prevent one-to-one back-translation from producing accurate paraphrase probabilities.

Translating from multiple pivot sentences into one target sentence requires that the decoder be re-defined. Firat et al. (2016b) propose several ways in which multiple pivot sentences can be incorporated into a NMT decoder. We extended their late averaging approach to incorporate weights. Consider the case of two pivot sentences from the same language, F_1 and F_2 . Each translation path individ-

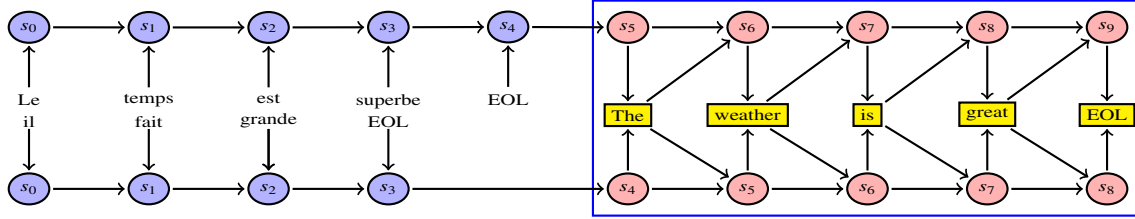


Figure 1: Late-weighted combination: two pivot sentences are simultaneously translated to one target sentence. Blue circles indicate the encoders, which individually encode the two source sentences. After the EOL token is seen, decoding starts (red circles). At each time step the two decoders produce a probability distribution over all words, which are then combined (in the yellow square) using Equation (6). From this combined distribution a word is chosen, which is then given as input to each decoder.

ually computes the distribution over the target vocabulary $P(y_{t'} = w|y_{<t'}, F_1)$ and $P(y_{t'} = w|y_{<t'}, F_2)$. Our *late-weighted combination* approach defines the path with respect to both translations as:

$$P(y_{t'} = w|y_{<t'}, F_1, F_2) = \lambda_1 P(y_{t'} = w|y_{<t'}, F_1) + \lambda_2 P(y_{t'} = w|y_{<t'}, F_2)$$

While Firat et al. (2016b) train a new model to capture these joint translations, we leave the model unchanged, instead treating PARANET as a *meta* encoder-decoder model (see Figure 1).

Unlike late averaging, PARANET assigns weights λ to each pivot sentence. These weights are set to the initial translation probabilities $P(F_i|E_1)$, thus capturing the model’s confidence in the accuracy of the translation:

$$P(y_{t'}=w|y_{<t'}, F_1, F_2) = P(F_1|E_1)P(y_{t'}=w|y_{<t'}, F_1) + P(F_2|E_1)P(y_{t'}=w|y_{<t'}, F_2)$$

Which can be trivially extended to include all translations from the K -best list:

$$P(y_{t'} = w|y_{<t'}, \mathcal{F}) = \frac{1}{\sum_{i=1}^K P(F_i|E_1)} \cdot P(y_{t'} = w|y_{<t'}, F_i) \quad (6)$$

To ensure a probability distribution, we normalize the K -best list \mathcal{F} , such that the translation probabilities sum to one.

Multi-lingual Pivoting PARANET further expands on the multi pivot approach by pivoting not only over multiple sentences from one language, but also over multiple sentences from multiple languages. Multi-lingual pivoting has been recently shown to improve translation quality (Firat et al., 2016b), especially for low-resource language pairs. Here, we hypothesize that it will also lead to more accurate paraphrases.

Multi-lingual pivoting requires a small extension to late-weighted combination. We illustrate with German as a second language. First,

the source sentence is translated into a K -best list of French \mathcal{F}^{Fr} , and a K -best list of German \mathcal{F}^{De} . Late-weighted combination is then applied, producing $P(y_{t'} = w|y_{<t'}, \mathcal{F}^{Fr})$ and $P(y_{t'} = w|y_{<t'}, \mathcal{F}^{De})$. These two output distributions are averaged, producing a multi-sentence, multi-lingual paraphrase probability:

$$P(y_{t'} = w|y_{<t'}, \mathcal{F}^{Fr}, \mathcal{F}^{De}) = \frac{1}{2} (P(y_{t'} = w|y_{<t'}, \mathcal{F}^{Fr}) + P(y_{t'} = w|y_{<t'}, \mathcal{F}^{De}))$$

which is used to obtain probability distributions over sentences:

$$P(E_2|E_1) = \prod_{t'=1}^{T_{E_2}} P(y_{t'}|y_{<t'}, \mathcal{F}^{Fr}, \mathcal{F}^{De}) \quad (7)$$

This can be trivially generalized to multiple languages. In this paper we use up to three.

3.3 PARANET Applications

The applications of PARANET are many and varied. We discuss some of these here and present detailed experimental evidence in Section 4. PARANET can be readily used for paraphrase detection (the task of analyzing two text segments and determining if they have the same meaning), by computing Equation (7). In addition, it can identify which linguistic units are considered paraphrases and to what extent. PARANET’s explanatory power stems from the attention mechanism inherent in the NMT systems.

In encoder-decoder models, attention is used during each step of decoding to indicate which are the relevant source words. In our case, each word of the paraphrase attends to words within the pivot sentence and each word in the pivot sentence attends to words within the source sentence. By summing out the weighted pivot sentence, it is possible to see the attention from paraphrase to

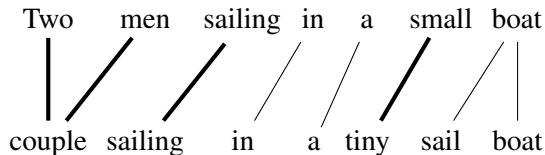


Figure 2: Attention between two sentences. Line thickness indicates the strength of the attention.

source:

$$\alpha(E_2^i, E_1^j, \mathcal{F}) = \sum_F P(E_2|E_1, F) \cdot \sum_m^{T_F} (\alpha_{i,m}^{E_2, F} \cdot \alpha_{m,j}^{F, E_1}) \quad (8)$$

An example shown in Figure 2 where attention has successfully identified the semantically equivalent parts of two sentences. Beyond providing interpretable paraphrasing, attention scores can be used as features in both generation and classification tasks.

Furthermore, PARANET can be readily used to perform text generation (via the NMT decoder) without additional resources or parameter estimation. It also learns phrase and sentence embeddings for free without any model adjustments or recourse to resources like PPDB.

4 Experiments

We evaluated PARANET in several ways: (a) we examined whether the paraphrases learned by our model correlate with human judgments of paraphrase quality; (b) we assessed PARANET in paraphrase and similarity detection tasks; and (c) in a sentence-level paraphrase generation task. We first present details on how PARANET and comparison models were trained and then discuss our results.

4.1 Neural Machine Translation Training

We used Groundhog¹ as the implementation of the NMT system for all experiments. We generally followed the settings and training procedure from previous work (Bahdanau et al., 2014; Sennrich et al., 2016a). As such, all networks have a hidden layer size of 1000, and an embedding layer size of 620. During training, we used Adadelta (Zeiler, 2012), a minibatch size of 80, and the training set was reshuffled between epochs. We trained a network for approximately 7 days on a single GPU, then the embedding layer was fixed and training continued, as suggested in Jean et al. (2015a), for 12 hours. Additionally, the softmax was calculated over a filtered list of candidate translations. Following Jean et al. (2015a), we set the common

¹github.com/sebastien-j/LV_groundhog

vocabulary size as 10000 and 25 uni-gram translations, using a bilingual dictionary based on fast-align (Dyer et al., 2013).

In our experiments, we used up to six encoder-decoder NMT models (three pairs); English→French, French→English, English→Czech, Czech→English, English→German, German→English. All systems were trained on the available training data from the WMT15 shared translation task (4.2 million, 15.7 million, and 39 million sentence pairs for EN↔DE, EN↔CS, and EN↔FR, respectively). For EN↔DE and EN→CS, we also had access to back-translated monolingual training data (Sennrich et al., 2016a), which we also used in training. The data was pre-processed using standard pre-processing scripts found in MOSES (Koehn et al., 2007). Rare words were split into sub-word units, following Sennrich et al. (2016b). BLEU scores for each NMT system can be seen in Table 1.

4.2 Statistical Machine Translation Training

Throughout our experiments we compare PARANET against a paraphrase model trained with a commonly used Statistical Machine Translation system (SMT), which we henceforth refer to as PARASTAT. Specifically, for each language pair used, an equivalent IBM Model 4 phrase-based translation model was trained. Additionally, an Operation Sequence Model (OSM) was included, which has been shown to improve the performance of SMT systems (Durrani et al., 2011). SMT translation models were implemented using both GIZA++ (Och and Ney, 2003) and MOSES (Koehn et al., 2007) and were trained using the same pre-processed bilingual data provided to the NMT systems. The SMT systems used a KenLM 5-gram language model (Heafield, 2011), trained on the mono-lingual data from WMT 2015. For all languages pairs, both KenLM and MOSES were trained using the standard settings. BLEU scores for the SMT systems are given in Table 1.

Under the SMT models, paraphrase probabilities were calculated analogously to Equation (7):

$$P(E_2|E_1, \mathcal{F}) = \sum_F^{\mathcal{F}} P(E_2|F)P(F|E_1) \quad (9)$$

where $P(E_2|F)$ and $P(F|E_1)$, are defined by the phrase based translation model, and \mathcal{F} denotes the K -best translations of E_1 , whose probabilities are normalized. Unlike PARANET these pivot sentences have to be combined outside of the decoder.

Direction	F→E		E→F	
System	SMT	NMT	SMT	NMT
French	0.241	0.201	0.233	0.271
German	0.207	0.282	0.208	0.248
Czech	0.216	0.197	0.145	0.176

Table 1: BLEU scores (WMT 2015 test set) for SMT and NMT models (foreign to English (F→E) and English to foreign (E→F) directions).

4.3 Correlation with Human Judgments

The PPDB 2.0 Human Evaluation data set is a sample of paraphrase pairs taken from PPDB which have been human annotated for semantic similarity (Pavlick et al., 2015). 26,455 samples were taken from range of syntactic categories, resulting in paraphrase candidates varying from single words to multi-word expressions. Each paraphrase pair was judged by five people on a 5-point scale. Ratings were then averaged giving each paraphrase pair a score between 1 and 5.

Using this dataset we measure the correlation (Spearman ρ) between (length normalized) PARANET probabilities (Equation (7)) assigned to paraphrase pairs and human judgments. Figure 3 shows correlation coefficients for all language pairs using a single foreign pivot and 200 pivots. Across all language combinations multiple pivots² achieve better correlations, with the German, Czech pair performing best with $\rho = 0.53$. For comparison, Pavlick et al. (2015) report a correlation of $\rho = 0.41$ using Equation (9) and PPDB (Ganitkevitch et al., 2013). The latter contains over 100 million paraphrases and was constructed over several English-to-foreign parallel corpora including Europarl v7 (Koehn, 2005) which contains bitexts for the 19 European languages.

Following Pavlick et al. (2015), we next developed a supervised scoring model. Specifically, we fit a decision tree regressor on the PPDB 2.0 dataset using the implementation provided in scikit-learn (Pedregosa et al., 2011). To improve accuracy and control over-fitting we built an ensemble of regression trees using the Extra-Trees algorithm (Geurts et al., 2006) which fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset. In our experiments 1,000 trees were trained to minimize mean square error. The regressor was trained with the following basic features: sentence length,

²Across tasks and datasets we find that multiple pivots outperform single pivots. We omit these comparisons from subsequent experiments for the sake of brevity.

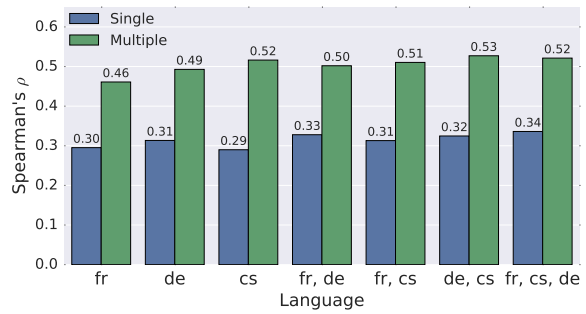


Figure 3: Correlation of PARANET predictions against human ratings for paraphrase pairs. Comparison using single and multiple pivots, across language combinations.

1-4 gram string similarity, the paraphrase probability $P(E_2|E_1)$, the language model score $P(E_1)$, cosine distance of the sentence vectors, as calculated by the encoder. To address the problem of rare sentences receiving low probabilities regardless of the source sentence, we create an inverse weighting by $P(E_2|E_2)$, which approximates how difficult it is to recover E_2 :

$$pscore(E_2, E_1) = \frac{P(E_2|E_1)}{P(E_2|E_1) + P(E_2|E_2)} \quad (10)$$

Two features reflect the alignment between candidate paraphrases. We built an alignment matrix according to Equation (8), and used the mean of the diagonal as feature. This acts as a proxy of how much movement there is between two paraphrases. The second feature is the number of unaligned words which we compute by calculating hard alignments between the two paraphrases.

Regressors varied with respect to how $P(E_2|E_1)$ was computed, keeping the string based features the same. Equations (7) and (9) were used to calculate paraphrase probability for PARANET and PARASTAT, respectively. For both models beam search (with width set to 100) was used to generate the K -best list. For each language, the K -best list is the union of the 100-best list of E_1 and the 100-best list of E_2 , giving a maximum of 200 pivot sentences. As set out in Pavlick et al. (2015) evaluation is done using cross validation: in each fold, we hold out 200 phrases. Table 2 presents results for PARANET and PARASTAT using different languages as pivots. PARANET outperforms PARASTAT across the board. Furthermore, despite using fewer features and pivot languages, it obtains a closer correspondence to human data compared to PPDB 2.0 (Pavlick et al., 2015).

Model	PARASTAT	PARANET
<i>fr</i>	0.574	0.700
<i>de</i>	0.638	0.710
<i>cz</i>	0.564	0.713
<i>de,fr</i>	0.566	0.722
<i>de,cz</i>	0.640	0.731
<i>fr,cz</i>	0.569	0.724
<i>fr, cz, de</i>	0.633	0.735
PPDB 2.0	0.713	

Table 2: Correlation (Spearman ρ) of supervised models against human ratings for paraphrase pairs. Boldface indicates the best performing model.

4.4 Paraphrase Identification and Similarity

The SemEval-2015 shared task on Paraphrase and Semantic Similarity In Twitter (PIT) uses a training and development set of 17,790 sentence pairs and a test set of 972 sentence pairs. By design, the dataset contains colloquial sentences representing informal language usage and sentence pairs which are lexically similar but semantically dissimilar. Sentence pairs were crawled from Twitter’s trending topics and associated tweets (see Xu et al. (2014) for details). The shared task consists of a (binary) paraphrase identification subtask (i.e., determine whether two sentences are paraphrases) and an optional semantic similarity task (i.e., determine the similarity between two sentences on a scale of 1–5, where 5 means completely equivalent and 1 not equivalent).

We trained a decision tree regressor on the PIT-2015 similarity dataset using the features described above. Once trained, the decision tree regressor can be readily applied to the semantic similarity subtask. For the paraphrase detection subtask, we use the same model and apply a threshold (optimized on the validation set) such that those pairs that are over this threshold are deemed paraphrases.

Tables 3 and 4 present our results on the two subtasks together with previously published results. We evaluate system performance on the detection task using F1 (the harmonic mean of precision and recall). For semantic similarity, system outputs are compared by Pearson correlation against human scores. The first block in the tables summarize results for PARANET and PARASTAT using different languages as pivots. The second block includes three baselines provided by the organizers of the shared task: a random baseline, a logistic regression baseline with minimal

Model	PARASTAT	PARANET
<i>fr</i>	0.613	0.624
<i>de</i>	0.616	0.620
<i>cz</i>	0.620	0.622
<i>de, fr</i>	0.602	0.622
<i>de, cz</i>	0.606	0.615
<i>fr, cz</i>	0.600	0.634
<i>fr, cz, de</i>	0.596	0.620
random	0.266	
WTMF	0.536	
logistic reg	0.589	
ASOBK	0.674	
MITRE	0.667	

Table 3: Paraphrase detection results (F1) on the PIT-2015 data set. Boldface indicates the best performing paraphrasing model.

Model	PARASTAT	PARANET
<i>fr</i>	0.540	0.569
<i>de</i>	0.543	0.571
<i>cz</i>	0.547	0.569
<i>de, fr</i>	0.543	0.569
<i>de, cz</i>	0.540	0.570
<i>fr, cz</i>	0.546	0.568
<i>fr, cz, de</i>	0.539	0.568
random	0.017	
WTMF	0.350	
logistic reg	0.511	
ASOBK	0.475	
MITRE	0.619	

Table 4: Semantic similarity results (Pearson) on the PIT-2015 data set. Boldface indicates the best performing paraphrasing model.

n-gram word overlap features; and a model which uses weighted matrix factorization (WTMF) and has access to dictionary definitions provided in WordNet, OntoNotes, and Wiktionary (Guo and Diab, 2012). The last two rows show the highest scoring systems: ASOBK (Eyecioglu and Keller, 2015) ranked 1st in the identification subtask and MITRE (Zarrella et al., 2015) in the similarity subtask. Whereas ASOBK uses knowledge-lean features based on word and character n-gram overlap, MITRE is a combination of multiple systems including mixtures of string matching metrics, alignments using tweet-specific word representations, and recurrent neural networks.

As can be seen, PARANET achieves better similarity and detection score than all baselines and PARASTAT, for any combinations of lan-

Model	PARASTAT	PARANET
<i>fr</i>	0.657	0.682
<i>de</i>	0.666	0.678
<i>cz</i>	0.649	0.688
<i>de, fr</i>	0.665	0.684
<i>de, cz</i>	0.662	0.687
<i>fr, cz</i>	0.654	0.690
<i>fr, cz, de</i>	0.658	0.689
Tokencos	0.587	
DLS@CU	0.801	

Table 5: Results on the Semeval-2015 semantic similarity dataset. Boldface indicates the best performing paraphrasing model.

guages. This is particularly impressive as the translation models were trained on very dissimilar data. Compared to the state of the art, PARANET fares worse, however our model was not particularly optimized on the PIT-2015 dataset which was merely used as a testbed for a fair comparison. It is thus reasonable to assume that taking into account more elaborate features (e.g., based on character embeddings) would improve performance. The highest semantic similarity score is obtained with PARANET trained using German data. The highest scoring paraphrase detection model was PARANET trained on French and Czech data. Interestingly, using multiple pivot languages seems to offer small improvements in most cases. The languages selected as pivots in our experiments were somewhat ad-hoc. We expect to get more mileage if these are selected from the same language family or with more linguistic insight (e.g., morphologically rich vs. poor).

4.5 Semantic Textual Similarity

In semantic textual similarity (STS), systems rate the degree of semantic equivalence between two text snippets. We present results on the Semeval-2015 English subtask which contains sentences from a wide range of domains, including newswire headlines, image descriptions, and answers from Q&A websites. The training/test sets consist of 11,250 and 3,000 sentence pairs, respectively. Sentence pairs are rated on a 1–5 scale, with 5 indicating they are completely equivalent.

We used the decision tree regressor with the same features described in the previous section. Again, we experimented with one, two, and three languages as pivots, and compared PARANET and PARASTAT directly. Our results are summarized in Table 5. The third block in the table presents a

simple cosine-based baseline provided by the organizers (Tokencos) and the top-performing system (DLS@CU) which uses PPDB paraphrases to identify semantically similar words and word2vec embeddings trained on approximately 2.8 billion tokens (Sultan et al., 2014).

PARANET outperforms PARASTAT on all languages and language combinations. Both systems outperform the Semeval baseline but are worse compared to the top scoring system. We see for PARANET Czech achieves the highest scores, this could be in part due to Czech non-strict word order, which allows paraphrases that are simple rearrangements not be penalized.

4.6 Paraphrase Generation

Finally, we evaluated PARANET (and PARASTAT) in a paraphrase generation task. We created sentential paraphrases for three (parallel monolingual) datasets representative of different domains and genres: (a) the Multiple-Translation Chinese (MTC) corpus (Huang et al., 2002) contains news stories from three sources of journalistic Mandarin Chinese text translated into English by 4 translation agencies; we sampled 1,000 sentences for training and testing, respectively (each source sentence had an average of 4 paraphrases); (b) the Jules Verne Twenty Thousand Leagues Under the Sea novel (Leagues) corpus (Pang et al., 2003) contains two English translations of the French novel; we sampled 500 sentences for training/testing (each source sentence had one paraphrase); and (c) the Wikianswers corpus (Fader et al., 2013) which contains questions taken from the website³ wiki answers; we sampled 1,000 questions for training/testing (each question has on average 21 paraphrases).

In order to select the best paraphrase candidate for a given input sentence, PARASTAT was optimized on the training set using Minimum Error Training (MERT, Och and Ney (2003)). MERT integrates automatic evaluation metrics such as BLEU into the training process to achieve optimal end-to-end performance. Naively optimizing for BLEU, however, will result in a trivial paraphrasing system heavily biased towards producing identity “paraphrases”. Sun and Zhou (2012) introduce iBLEU which we also adopt. iBLEU penalizes paraphrases which are similar to the source

³<http://wiki.answers.com/>

Model	PARASTAT	PARANET
<i>fr</i>	0.280	0.299
<i>de</i>	0.282	0.295
<i>cz</i>	0.280	0.291
Gold	0.599	

Table 6: Mean iBLEU across three datasets.

sentence and rewards those close to the target:

$$iBLEU(s, r_s, c) = \alpha BLEU(c, r_s) - (1 - \alpha) BLEU(c, s)$$

where s , is the source sentence, r_s , is the target and c is the candidate paraphrase. $(1 - \alpha)BLEU(c, s)$, measures the originality of the candidate paraphrase, $BLEU(c, r_s)$ measures semantic adequacy, and α is a tuning parameter which balances the two. Sentence level BLEU is calculated using plus one smoothing (Lin and Och, 2004).

PARANET relies on a relatively simple architecture which is trained end-to-end with the objective of maximizing the likelihood of the training data. Since evaluation metrics cannot be straightforwardly integrated into this training procedure, we reranked the k -best paraphrases obtained from PARANET using a simple classifier which favors sentences which are dissimilar to the source. Specifically, we trained a decision tree regression model with iBLEU as the target variable using the same features described in Section 4.4. Examples of paraphrases generated by PARANET are shown in the Appendix.

System output was assessed automatically using iBLEU with human-written paraphrases as reference. In addition, we evaluated the generated text by eliciting human judgments via Amazon Mechanical Turk. We randomly selected 100 source sentences from each data set and generated output with PARANET and PARASTAT (using German as a pivot). We also included a randomly selected human paraphrase as a goldstandard. Workers (self-reported native English speakers) were asked to rank the three paraphrases from best to worst (ties were allowed) in order of semantic equivalence (does the paraphrase convey the same meaning as the source?) and fluency (is the description written in well-formed English?). Participants were explicitly told to give high ranks to output demonstrating a fair amount of paraphrasing and low ranks to trivial paraphrases (e.g., deletion of articles or punctuation). We collected 5 responses per input sentence.

Table 6 summarizes our results across the three

Model	Wikianswers	Leagues	MTC	All
PARASTAT	2.09	2.38	2.23	2.26
PARANET	1.86	1.94	1.70	1.83
Humans	2.17	1.81	2.0	2.0

Table 7: Mean Rankings given to paraphrases by human participants (a lower score is better).

datasets. For the sake of brevity, we only show results with one pivot language since combinations performed slightly worse for both models. We set $\alpha = 0.8$ for iBLEU as we experimentally found it offers the best trade-off between semantic equivalence and dissimilarity. As an upper-bound we also measure iBLEU amongst the gold paraphrases provided by humans. Again, we observe that PARANET has a slight advantage over PARASTAT in terms of iBLEU, however both systems tend to paraphrase less compared to the gold-standard. Table 7 shows the mean ranks given to these systems by human subjects. An Analysis of Variance (ANOVA) revealed a reliable effect of system type. Post-hoc Tukey tests showed that PARANET is significantly ($p < 0.01$) better than PARASTAT across datasets; PARANET is also significantly ($p < 0.01$) better than the the gold standard on both MTC and the Wikianswers dataset. We attribute this to the noisy nature of these two datasets which contain a wealth of paraphrases, a few of which are ungrammatical, contain typos or abbreviations leading to low scores among humans.

5 Conclusions

In this work we presented PARANET, a neural paraphrasing model based on bilingual pivoting. Experimental results across several tasks (similarity prediction, paraphrase identification, and paraphrase generation) show that PARANET outperforms conventional paraphrasing methods. In the future, we plan to exploit the attention scores more directly for extracting paraphrase pairs (in analogy to PPDB) and as features for classification tasks (e.g., textual entailment). We would also like to investigate how PARANET can be adapted using reinforcement learning (Ranzato et al., 2016) to text generation tasks such as simplification and sentence compression.

Acknowledgments The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1; Mallinson) and the European Research Council (award number 681760; Lapata).

Appendix

Tables 8–10 show examples of PARANET output on the Wikianswers, Leagues, and MTC datasets.

Wikianswers
a. How many calories in a handful of strawberries? b. The number of calories in a handful of strawberries.
a. Beauty is not in the eye of the beholder. b. Beauty is not in the mind of the viewer.
a. What is the importance of employee satisfaction in an organization? b. What is the significance of staff satisfaction at an organisation?
a. What is the difference between electrical power and electrical energy? b. What is the difference between electrical energy and electrical power?
a. How many high tides happen at a given coast in any 24 hour period? b. How many high tides occur on a certain coast in 24 hours?
a. What is a beverage that starts with the letter p? b. What is a drink that begins with the letter p?
a. What Swiss mathematician and teacher was responsible for instituting the use of the symbol for π in mathematical notation? b. What Swiss mathematicians and teachers were responsible for the introduction of the symbol for π in math notation?
a. How do you make a pina colada? b. How do you do a Pina colada?
a. What is the difference between a captain and a skipper? b. What is the difference between being a captain and skipper?

Table 8: Sentences marked (a) are the input and (b) are PARANET paraphrases.

Leagues
a. "Faith i should never have believed it," said Conseil. b. "Faith, I never would have believed", Conseil said.
a. "I owed myself this revenge!" Said the Captain to the Canadian. b. "I am indebted to this revenge!" the captain told the Canadian.
a. "Well, sir, you will only get your deserts." b. "Well, sir, you are only getting your deserts."
a. "That's what I've been telling you Ned." b. "That's what I said, Ned."
a. Very much embarrassed, after having vainly exhausted our speaking resources, I knew not what part to take, when Conseil said: "if master will permit me I will relate it in German." b. It was very embarrassing that I had used up our speaking time, and I did not know what to do, as Conseil said: "If the Masters allow me, I shall refer to German."
a. Almost every day the panels in the lounge were open for some hours, and our eyes never tired of probing the mysteries of the underwater world. b. Almost every day, the panels opened in the lounge for a few hours, and our eyes never tired, the secrets of the underwater world.
a. I bowed, and desired to be conducted to the cabin destined for me. b. I bow to and wish I headed to the cabin for me.
a. I had one question left to address to Captain Nemo. b. I had a question left to Captain Nemo.
a. "I have not the foggiest notion, Professor Aronnax." b. I have no idea, Professor Aronnax.

Table 9: Sentences marked (a) are the input and (b) are PARANET paraphrases.

MTC
a. China expresses strong dissatisfaction over the Japanese leader's move this time. b. China expresses a strong dissatisfaction over Japanese leader's move.
a. We will accelerate the drafting of telecommunications legalization, amend the law of post and the regulations governing wireless telecommunications. b. We will speed up the design of telecommunications, change the law and regulations governing wireless telecommunication.
a. Liu said: the poverty-stricken areas are badly hit in the first stage of this year's floods and many counties and cities are listed as the poorest ones in the country. b. Liu said: poverty-stricken areas are hit hard in the first phase of this year's flooding and many counties and towns are listed as the poorest in the country.
a. (London, AP) The British government is working on resolving the increasingly serious problems of street crimes and will strengthen patrolling police. b. London, AP The British government is working to resolve the increasingly serious problems of street crime and will strengthen patrols.
a. Kida said that the dead killed by the heat wave were mostly old people with heart diseases. b. Kida said the dead by heatwave were mostly old people with heart disease.
a. Growth of Mobile Phone users in Mainland China to Slow Down. b. Growth of Mobile Phone users in Mainland China on Slow Down.
a. The survey report said that in the first six months of last year 18 sandstorms struck Beijing and they all originated from Inner Mongolia where 60% of the land is desert. b. The report said that 18 sandstorms struck Beijing in the first six months of last year, and they were all from Inner Mongolia, where 60% of the desert is desert.
a. The World Cup co-host by Japan and South Korea, will inaugurate on May 31. b. The World Cup, co-host Japan and South Korea, will open on May 31.
a. Two days ago, President Bush seemed opposed to this idea when he held talks with Sharon. b. Two days ago President Bush opposed this idea when he talks to Sharon.
a. Russia Faces Population Crisis. b. Russia's demographics problem.
a. Computer Crimes Cost US billions of Dollars Last Year. b. Computer Crimes Cost American Billions of Dollars.
a. However, many sports associations in Chile hope to cooperate with China not just for the table tennis alone. b. However, many sports federations in Chile are hoping to collaborate with China, not only for the table tennis players.

Table 10: Sentences marked (a) are the input and (b) are PARANET paraphrases.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.
- Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Associa-*

- tion for *Computational Linguistics*, pages 597–604, Ann Arbor, Michigan.
- Regina Barzilay. 2003. *Information Fusion for Multi-Document Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University.
- Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland.
- Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 17–24, New York City, USA.
- Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 196–205, Honolulu, Hawaii.
- Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China.
- Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1045–1054, Portland, Oregon, USA.
- Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia.
- Asli Eyecioğlu and Bill Keller. 2015. Twitter paraphrase identification with simple overlap features and svms. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 64–69, Denver, Colorado.
- Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria.
- Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California.
- Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. *CoRR*, abs/1606.04164.
- Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *The 9th edition of the Language Resources and Evaluation Conference*, Reykjavik, Iceland.
- Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1168–1179, Edinburgh, Scotland, UK.
- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia.
- Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.
- Weiwei Guo and Mona Diab. 2012. A simple unsupervised latent semantics based approach for sentence similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 586–590.
- Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197.
- Shudong Huang, David Graff, George Doddington, Linguistic Data Consortium, et al. 2002. *Multiple-translation Chinese corpus*. Linguistic Data Consortium, University of Pennsylvania.
- Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015a. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China.

- Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015b. Montreal neural machine translation systems for wmt15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal.
- Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA.
- Martin Kay. 1997. The proper place of men and machines in language translation. *Machine Translation*, 12(1-2):3–23.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, Sapporo, Japan.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.
- Philip Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86, Phuket, Thailand.
- Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics, Main Volume*, pages 605–612, Barcelona, Spain.
- Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):342–360.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.
- Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.
- Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 120–127, Prague, Czech Republic.
- Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
- Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 102–109, Edmonton, Canada.
- Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremb. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.
- Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471, Prague, Czech Republic.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA.
- Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. DLS@CU: Sentence similarity from word alignment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 241–246.
- Hong Sun and Ming Zhou. 2012. Joint learning of a dual smt system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–42, Jeju Island, Korea.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112. Curran Associates, Inc.
- Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491, Rochester, New York.
- John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association of Computational Linguistics – Volume 3, Issue 1*, pages 345–358.
- John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of 4th International Conference on Learning Representations*, San Juan, Puerto Rico.
- Kristian Woodsend and Mirella Lapata. 2014. Text rewriting improves semantic role labeling. *Journal of Artificial Intelligence Research*, 51:133–164.
- Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.
- Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.
- Guido Zarrella, John Henderson, Elizabeth M. Merkhofer, and Laura Strickhart. 2015. Mitre: Seven systems for semantic similarity in tweets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 12–17, Denver, Colorado.
- Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.
- Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Proceedings of ACL-08: HLT*, pages 780–788, Columbus, Ohio.
- Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California.

Multilingual Training of Crosslingual Word Embeddings

Long Duong,¹ Hiroshi Kanayama,² Tengfei Ma,³ Steven Bird^{1,4} and Trevor Cohn¹

¹Department of Computing and Information Systems, University of Melbourne

²IBM Research – Tokyo

³IBM T.J. Watson Research Center

⁴International Computer Science Institute, University of California Berkeley

Abstract

Crosslingual word embeddings represent lexical items from different languages using the same vector space, enabling crosslingual transfer. Most prior work constructs embeddings for a pair of languages, with English on one side. We investigate methods for building high quality crosslingual word embeddings for many languages in a unified vector space. In this way, we can exploit and combine information from many languages. We report competitive performance on bilingual lexicon induction, monolingual similarity and crosslingual document classification tasks.

1 Introduction

Monolingual word embeddings have facilitated advances in many natural language processing tasks, such as natural language understanding (Collobert and Weston, 2008), sentiment analysis (Socher et al., 2013), and dependency parsing (Dyer et al., 2015). Crosslingual word embeddings represent words from several languages in the same low dimensional space. They are helpful for multilingual tasks such as machine translation (Brown et al., 1993) and bilingual named entity recognition (Wang et al., 2013). Crosslingual word embeddings can also be used in transfer learning, where the source model is trained on one language and applied directly to another language; this is suitable for the low-resource scenario (Yarowsky and Ngai, 2001; Duong et al., 2015b; Das and Petrov, 2011; Täckström et al., 2012).

Most prior work on building crosslingual word embeddings focuses on a pair of languages. English is usually on one side, thanks to the wealth

of available English resources. However, it is highly desirable to have a crosslingual word embeddings for many languages so that different relations can be exploited.¹ For example, since Italian and Spanish are similar, they are excellent candidates for transfer learning. However, few parallel resources exist between Italian and Spanish for directly building bilingual word embeddings. Our multilingual word embeddings, on the other hand, map both Italian and Spanish to the same space without using any direct bilingual signal between them. In addition, multilingual word embeddings allow multiple source language transfer learning, producing a more general model and overcoming data sparseness (McDonald et al., 2011; Guo et al., 2016; Agić et al., 2016). Moreover, multilingual word embeddings are also crucial for multilingual applications such as multi-source machine translation (Zoph and Knight, 2016), and multi-source transfer dependency parsing (McDonald et al., 2011; Duong et al., 2015a).

We propose several algorithms to map bilingual word embeddings to the same vector space, either during training or during post-processing. We apply a linear transformation to map the English side of each pretrained crosslingual word embedding to the same space. We also extend Duong et al. (2016), which used a lexicon to learn bilingual word embeddings. We modify the objective function to jointly build multilingual word embeddings during training. Unlike most prior work which focuses on downstream applications, we measure the quality of our multilingual word embeddings in three ways: bilingual lexicon induction, monolingual word similarity, and crosslingual document classification tasks. Relative to a benchmark of

¹From here on we refer to crosslingual word embeddings for a pair of languages and multiple languages as *bilingual word embeddings* and *multilingual word embeddings* respectively.

training on each language pair separately and to various published multilingual word embeddings, we achieved high performance for all the tasks.

In this paper we make the following contributions: (a) novel algorithms for post hoc combination of multiple bilingual word embeddings, applicable to any pretrained bilingual model; (b) a method for jointly learning multilingual word embeddings, extending Duong et al. (2016), to jointly train over monolingual corpora in several languages; (c) achieving competitive results in bilingual, monolingual and crosslingual transfer settings.

2 Related work

Crosslingual word embeddings are typically based on co-occurrence statistics from parallel text (Luong et al., 2015; Gouws et al., 2015; Chandar A P et al., 2014; Klementiev et al., 2012; Kočiský et al., 2014; Huang et al., 2015). Other work uses more widely available resources such as comparable data (Vulić and Moens, 2015) and shared Wikipedia entries (Søgaard et al., 2015). However, those approaches rely on data from Wikipedia, and it is non-trivial to extend them to languages that are not covered by Wikipedia. Lexicons are another source of bilingual signal, with the advantage of high coverage. Multilingual lexical resources such as PanLex (Kamholz et al., 2014) and Wiktionary² cover thousands of languages, and have been used to construct high performance crosslingual word embeddings (Mikolov et al., 2013a; Xiao and Guo, 2014; Faruqui and Dyer, 2014).

Previous work mainly focuses on building word embeddings for a pair of languages, typically with English on one side, with the exception of Coulmance et al. (2015), Søgaard et al. (2015) and Ammar et al. (2016). Coulmance et al. (2015) extend the bilingual skipgram model from Luong et al. (2015), training jointly over many languages using the Europarl corpora. We also compare our models with an extension of Huang et al. (2015) adapted for multiple languages also using bilingual corpora. However, parallel data is an expensive resource and using parallel data seems to under-perform on the bilingual lexicon induction task (Vulić and Moens, 2015). While Coulmance et al. (2015) use English as the pivot language, Søgaard et al. (2015) learn multilingual word em-

²wiktionary.org

beddings for many languages using Wikipedia entries which are the same for many languages. However, their approach is limited to languages covered in Wikipedia and seems to under-perform other methods. Ammar et al. (2016) propose two algorithms, MultiCluster and MultiCCA, for multilingual word embeddings using set of bilingual lexicons. MultiCluster first builds the graph where nodes are lexical items and edges are translations. Each cluster in this graph is an anchor point for building multilingual word embeddings. MultiCCA is an extension of Faruqui and Dyer (2014), performing canonical correlation analysis (CCA) for multiple languages using English as the pivot. A shortcoming of MultiCCA is that it ignores polysemous translations by retaining only one-to-one dictionary pairs (Gouws et al., 2015), disregarding much information. As a simple solution, we propose a simple post hoc method by mapping the English parts of each bilingual word embedding to each other. In this way, the mapping is always exact and one-to-one.

Duong et al. (2016) constructed bilingual word embeddings based on monolingual data and PanLex. In this way, their approach can be applied to more languages as PanLex covers more than a thousand languages. They solve the polysemy problem by integrating an EM algorithm for selecting a lexicon. Relative to many previous crosslingual word embeddings, their joint training algorithm achieved state-of-the-art performance for the bilingual lexicon induction task, performing significantly better on monolingual similarity and achieving a competitive result on cross lingual document classification. Here we also adopt their approach, and extend it to multilingual embeddings.

2.1 Base model for bilingual embeddings

We briefly describe the base model (Duong et al., 2016), an extension of the continuous bag-of-word (CBOW) model (Mikolov et al., 2013a) with negative sampling. The original objective function is

$$\sum_{i \in D} \left(\log \sigma(\mathbf{u}_{w_i}^\top \mathbf{h}_i) + \sum_{j=1}^p \log \sigma(-\mathbf{u}_{w_{i_j}}^\top \mathbf{h}_i) \right), \quad (1)$$

where D is the training data, $\mathbf{h}_i = \frac{1}{2k} \sum_{j=-k; j \neq 0}^k \mathbf{v}_{w_{i+j}}$ is a vector encoding the context over a window of size k centred around position i , \mathbf{V} and $\mathbf{U} \in \mathbb{R}^{|V_e| \times d}$ are learned matrices referred to as the context and centre word

embeddings, where V_e is the vocabulary and p is the number of negative examples randomly drawn from a noise distribution, $w_{ij} \sim P_n(w)$.

Duong et al. (2016) extend the CBOW model for application to two languages, using monolingual text in both languages and a bilingual lexicon. Their approach augments CBOW by generating not only the middle word, but also its translation in the other language. This is done by first selecting a translation \bar{w}_i from the lexicon for the middle word w_i , based on the cosine distance between the context h_i and the context embeddings \mathbf{V} for each candidate foreign translation. In this way source monolingual training contexts must generate both source and target words, and similarly target monolingual training contexts also generate source and target words. Overall this results in compatible word embeddings across the two languages, and highly informative nearest neighbours across the two languages. This leads to the new objective function

$$\sum_{i \in D_s \cup D_t} \left(\log \sigma(\mathbf{u}_{w_i}^\top \mathbf{h}_i) + \log \sigma(\mathbf{u}_{\bar{w}_i}^\top \mathbf{h}_i) \right) + \sum_{j=1}^p \log \sigma(-\mathbf{u}_{w_{ij}}^\top \mathbf{h}_i) + \delta \sum_{w \in V_s \cup V_t} \|\mathbf{u}_w - \mathbf{v}_w\|_2^2, \quad (2)$$

where D_s and D_t are source and target monolingual data, V_s and V_t are source and target vocabulary. Comparing with the CBOW objective function in Equation (1), this represents two additions: the translation cross entropy $\log \sigma(\mathbf{u}_{\bar{w}_i}^\top \mathbf{h}_i)$, and a regularisation term $\sum_{w \in V_s \cup V_t} \|\mathbf{u}_w - \mathbf{v}_w\|_2^2$ which penalises divergence between context and center word embedding vectors for each word type, which was shown to improve the embedding quality (Duong et al., 2016).

3 Post hoc Unification of Embeddings

Our goal is to learn multilingual word embeddings over more than two languages. One simple way to do this is to take several learned bilingual word embeddings which share a common target language (here, English), and map these into a shared space (Mikolov et al., 2013a; Faruqui and Dyer, 2014). In this section we propose post hoc methods, however in §4 we develop an integrated multilingual method using joint inference.

Formally, the input to the post hoc combination methods are a set of n pre-trained bilingual word

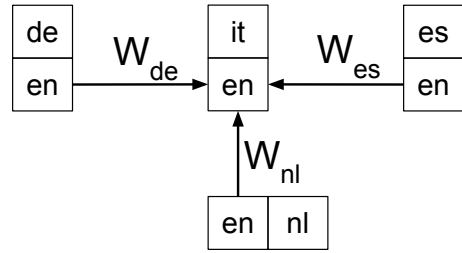


Figure 1: Examples of unifying four bilingual word embeddings between en and it, de, es, nl to the same space using post hoc linear transformation.

embedding matrices, i.e., $C_i = \{(E_i, F_i)\}$ with $i \in \mathbf{F}$ is the set of foreign languages (not English), $E_i \in \mathbb{R}^{|V_{e_i}| \times d}$ are the English word embeddings and $F_i \in \mathbb{R}^{|V_{f_i}| \times d}$ are foreign language word embeddings for language i , with V_{e_i} and V_{f_i} being the English and foreign language vocabularies and d is the embedding dimension. These bilingual embeddings can be produced by any method, e.g., those discussed in §2.

Linear Transformation. The simplest method is to learn a linear transformation which maps the English part of each bilingual word embedding into the same space (inspired by Mikolov et al. (2013a)), as illustrated in Figure 1. One language pair is chosen as the pivot, en-it in this example, and the English side of the other language pairs, en-de, en-es, en-nl, are mapped to closely match the English side of the pivot, en-it. This is achieved through learning linear transformation matrices for each language, W_{de}, W_{es} and W_{nl} , respectively, where each $W_i \in \mathbb{R}^{d \times d}$ is learned to minimize the objective function $\|E_i \times W_i - E_{pivot}\|_2^2$ where E_{pivot} is the English embedding of the pivot pair, en-it.

Each foreign language f_i is then mapped to the same space using the learned matrix W_i , i.e., $F'_i = F_i \times W_i$. These projected foreign embeddings are then used in evaluation, along with the English side of the language pair with largest English vocabulary coverage, i.e., biggest $|V_{e_i}|$. Together these embeddings allow for querying of monolingual and cross-lingual word similarity, and multilingual transfer of trained models.

The advantage of this approach is that it is very fast and simple to train, since the objective function is strictly convex and has a closed form solution. Moreover, unlike Mikolov et al. (2013a) who learn the projection from a source to a target

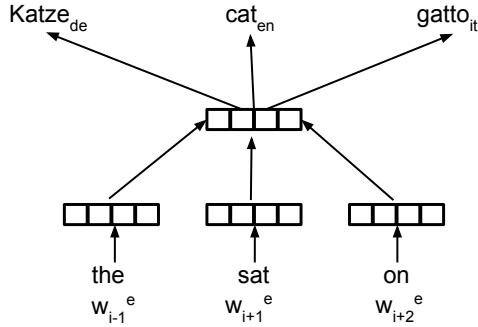


Figure 2: Examples of our multilingual joint training model without mapping for learning multilingual embeddings for three languages en , it , de using joint inference.

language, we learn the projection from English to English, thus do not require a lexicon, sidestepping the polysemy problem.³

4 Multilingual Joint Training

Instead of combining bilingual word embeddings in the post-processing step, it might be more beneficial to do it during training, so that languages can interact with each other more freely. We extend the method in §2.1 to jointly learn the multilingual word embeddings during training. The input to the model is the combined monolingual data for each language and the set of lexicons between any language pair.

We modify the base model (Duong et al., 2016) to accommodate more languages. For the first step, instead of just predicting the translation for a single target language, we predict the translation for all languages in the lexicon. That is, we compute $w_i^f = \operatorname{argmax}_{w \in \operatorname{dict}_e^f(w_i^e)} \cos(\mathbf{v}_w, \text{context})$, which is the best translation in language f of source word w_i^e in language e , given the bilingual lexicon dict_e^f and the context. For the second step, we jointly predict word w_i^e and all translations w_i^f in all foreign languages $f \in \mathbf{T}$ that we have dictionary dict_e^f as illustrated in Figure 2.

³A possible criticism of this approach is that a linear transformation is not powerful enough for the required mapping. We experimented with non-linear transformations but did not observe any improvements. Faruqui and Dyer (2014) extended Mikolov et al. (2013a) as they projected both source and target languages to the same space using canonical correlation analysis (CCA). We also adopted this approach for multilingual environment by applying multi-view CCA to map the English part of each pre-trained bilingual word embedding to the same space. However, we only observe minor improvements.

The English word *cat* might have several translations in German $\{Katze, Raupe, Typ\}$ and Italian $\{gatto, gatta\}$. In the first step, we select the closest translation given the context for each language, i.e. *Katze* and *gatto* for German and Italian respectively. In the second step, we jointly predict the English word *cat* together with selected translations *Katze* and *gatto* using the following modified objective function:

$$\mathcal{O} = \sum_{i \in D_{all}} \left(\log \sigma(\mathbf{u}_{w_i^e}^\top \mathbf{h}_i) + \sum_{f \in \mathbf{T}} \log \sigma(\mathbf{u}_{w_i^f}^\top \mathbf{h}_i) + \sum_{j=1}^p \log \sigma(-\mathbf{u}_{w_{ij}^e}^\top \mathbf{h}_i) \right) + \delta \sum_{w \in V_{all}} \|\mathbf{u}_w - \mathbf{v}_w\|_2^2, \quad (3)$$

where D_{all} and V_{all} are the combined monolingual data and vocabulary for all languages. Each of the p negative samples, w_{ij} , are sampled from a unigram model over the combined vocabulary V_{all} .

Explicit mapping. As we keep adding more languages to the model, the hidden layer in our model – shared between all languages – might not be enough to accommodate all languages. However, we can combine the strength of the linear transformation proposed in §3 to our joint model as described in Equation (3). We explicitly learn the linear transformation jointly during training by adding the following regularization term to the objective function:

$$\mathcal{O}' = \mathcal{O} + \alpha \sum_{i \in D_e} \sum_{f \in \mathbf{F}} \|\mathbf{u}_{w_i^f} W_f - \mathbf{u}_{w_i^e}\|_2^2, \quad (4)$$

where D_e is the English monolingual data (since we use English as the pivot language), \mathbf{F} is the set of foreign languages (not English), $W_f \in \mathbb{R}^{d \times d}$ is the linear transformation matrix, and α controls the contribution of the regularization term and will be tuned in §6.⁴ Thus, the set of learned parameters for the model are the word and context embeddings \mathbf{U}, \mathbf{V} and $|\mathbf{F}|$ linear transformation matrices, $\{W_f\}_{f \in \mathbf{F}}$. After training is finished, we linearly transform the foreign language embeddings with the corresponding learned matrix W_f , such that all embeddings are in the same space.

5 Experiment Setup

Our experimental setup is based on that of Duong et al. (2016). We use the first 5 million sen-

⁴For an efficient implementation, we apply this constraint to only 10% of English monolingual data.

	Model	it-en		es-en		nl-en		nl-es		Average	
		rec ₁	rec ₅	rec ₁	rec ₅	rec ₁	rec ₅	rec ₁	rec ₅	rec ₁	rec ₅
Baselines	MultiCluster	35.6	64.3	34.9	62.5	-	-	-	-	-	-
	MultiCCA	63.4	77.3	58.5	72.7	-	-	-	-	-	-
	MultiSkip	57.6	68.5	49.3	58.9	-	-	-	-	-	-
	MultiTrans	72.1	83.1	71.5	82.2	-	-	-	-	-	-
Ours	Linear	78.5	88.2	69.3	81.8	74.9	87.0	66.3	79.7	72.2	84.2
	Joint	79.4	89.7	73.6	84.6	76.6	89.6	69.4	82.0	74.7	86.5
	+ Mapping	81.6	90.5	74.6	87.4	77.9	91.4	71.6	83.5	76.4	88.2
	BiWE	80.8	90.4	74.7	85.4	79.1	90.5	71.7	80.7	76.6	86.7

Table 1: Bilingual lexicon induction performance for four pairs. Bilingual word embeddings (BiWE) is the state-of-the-art result from Duong et al. (2016) where each pair is trained separately. Our proposed methods including linear transformation (Linear), joint prediction as in Equation (3) (Joint) and joint prediction with explicit mapping as in Equation (4) (+mapping). We report recall at 1 and 5 with respect to four baseline multilingual word embeddings. The best scores for are shown in bold.

tences from the tokenized monolingual data from the Wikipedia dump from Al-Rfou et al. (2013).⁵ The dictionary is from PanLex which covers more than 1,000 language varieties. We build multilingual word embeddings for 5 languages (*en*, *it*, *es*, *nl*, *de*) jointly using the same parameters as Duong et al. (2016).⁶ During training, for a fairer comparison, we only use lexicons between English and each target language. However, it is straightforward to incorporate a lexicon between any pair of languages into our model. The pre-trained bilingual word embeddings for the post-processing experiment in §3 are also from Duong et al. (2016).

In the following sections, we evaluate the performance of our multilingual word embeddings in comparison with bilingual word embeddings and previous published multilingual word embeddings (MultiCluster, MultiCCA, MultiSkip and MultiTrans) for three tasks: bilingual lexicon induction (§6), monolingual similarity (§7) and crosslingual document classification (§8). MultiCluster and MultiCCA are the models proposed from Ammar et al. (2016) trained on monolingual data using bilingual lexicons extracted from aligning Europarl corpus. MultiSkip is the reimplementation of the multilingual skipgram model from Coul-

mance et al. (2015). MultiTrans is the multilingual version of the translation invariance model from Huang et al. (2015). Both MultiSkip and MultiTrans are trained directly on parallel data from Europarl. All the previous work is trained with 512 dimensions on 12 languages acquired directly from Ammar et al. (2016).

6 Bilingual Lexicon Induction

In this section we evaluate our multilingual models on the bilingual lexicon induction (BLI) task, which tests the bilingual quality of the model. Given a word in the source language, the model must predict the translation in the target language. We report recall at 1 and 5 for the various models listed in Table 1. The evaluation data for *it-en*, *es-en*, and *nl-en* pairs was manually constructed (Vulić and Moens, 2015). We extend the evaluation for *nl-es* pair which do not involve English.⁷

The BiWE results for pairs involving English in Table 1 are from Duong et al. (2016), the current state of the art in this task. For the *nl-es* pair, we cannot build bilingual word embeddings, since we do not have a corresponding bilingual lexicon. Instead, we use English as the pivot language. To get the *nl-es* translation, we use two bilingual embeddings of *nl-en* and *es-en* from Duong et al. (2016). We get the best English translation for the Dutch word, and get the top 5 Spanish

⁵We will use the whole data if there are less than 5 million sentences.

⁶Default learning rate of 0.025, negative sampling with 25 samples, subsampling rate of value $1e^{-4}$, embedding dimension $d = 200$, window size 48, run for 15 epochs and $\delta = 0.01$ for combining word and context embeddings.

⁷We build 1,000 translation pairs for *nl-es* pair with the source word from Vulić and Moens (2015) and ground truth candidates from Google Translate but manually verified.

translations with respect to the English word. This simple trick performs surprisingly well, probably because bilingual word embeddings involving English such as `nl-en` and `es-en` from Duong et al. (2016) are very accurate.

For the linear transformation, we use the first pair `it-en` as the pivot and learn to project `es-en`, `de-en`, `nl-en` pairs to this space as illustrated in Figure 1. We use English part ($E'_{biggest}$) from transformed `de-en` pair as the English output. Despite simplicity, linear transformation performs surprisingly well.

Our joint model to predict all target languages simultaneously, as described in Equation (3), performs consistently better in contrast with linear transformation at all language pairs. The joint model with explicit mapping as described in Equation (4) can be understood as the combination of joint model and linear transformation. For this model, we need to tune α in Equation (4). We tested α with value in range $\{10^{-i}\}_{i=0}^5$ using `es-en` pair on BLI task. $\alpha = 0.1$ gives the best performance. To avoid over-fitting, we use the same value of α for all experiments and all other pairs. With this tuned value α , our joint model with mapping clearly outperforms other proposed methods on all pairs. More importantly, this result is substantially better than all the baselines across four language pairs and two evaluation metrics. Comparing with the state of the art (BiWE), our final model (joint + mapping) are more general and more widely applicable, however achieves relatively better result, especially for recall at 5.

7 Monolingual similarity

The multilingual word embeddings should preserve the monolingual property of the languages. We evaluate using the monolingual similarity task proposed in Luong et al. (2015). In this task, the model is asked to give the similarity score for a pair of words in the same language. This score is then measured against human judgment. Following Duong et al. (2016), we evaluate on three datasets, WordSim353 (WS-en), RareWord (RW-en), and the German version of WordSim353 (WS-de) (Finkelstein et al., 2001; Luong et al., 2013; Luong et al., 2015).

Table 2 shows the result of our multilingual word embeddings with respect to several baselines. The trend is similar to the bilingual lexicon induction task. Linear transformation per-

	Model	WS-de	WS-en	RW-en
Baselines	MultiCluster	51.0 [98.3]	53.9 [100]	38.1 [57.6]
	MultiCCA	60.2 [99.7]	66.3 [100]	43.1 [71.1]
	MultiSkip	48.4 [96.6]	51.2 [99.7]	33.9 [55.4]
	MultiTrans	56.4 [92.6]	61.1 [97.2]	51.1 [23.1]
Ours	Linear	67.5 [99.4]	74.7 [100]	45.4 [75.5]
	Joint	68.5 [99.4]	74.6 [100]	43.8 [75.5]
	Joint + Mapping	70.4 [99.4]	74.4 [100]	45.1 [75.5]
	BiWE	71.1 [99.4]	76.2 [100]	44.0 [75.5]

Table 2: Spearman’s rank correlation for monolingual similarity measurement for various models on 3 datasets WS-de (353 pairs), WS-en (353 pairs) and RW-en (2034 pairs). We compare against 4 baseline multilingual word embeddings. BiWE is the result from Duong et al. (2016) where each pair is trained separately which serves as the reference for the best bilingual word embeddings. The best results for multilingual word embeddings are shown in bold. Numbers in square brackets are the coverage percentage.

forms surprisingly well. Our joint model achieves a similar result, with linear transformation (better on WS-de but worse on WS-en and RW-en). Our joint model with explicit mapping regains the drop and performs slightly better than linear transformation. More importantly, this model is substantially better than all baselines, except for MultiTrans on RW-en dataset. This can probably be explained by the low coverage of MultiTrans on this dataset. Our final model (Joint + Mapping) is also close to the best bilingual word embeddings (BiWE) performance reported by Duong et al. (2016).

8 Crosslingual Document Classification

In the previous sections, we have shown that our methods for building multilingual word embeddings, either in the post-processing step or during training, preserved high quality bilingual and monolingual relations. In this section, we demonstrate the usefulness of multi-language crosslingual word embeddings through the crosslingual document classification (CLDC) task.

This task exploits transfer learning, where the document classifier is trained on the source language and tested on the target language. The source language classifier is transferred to the target language using crosslingual word embeddings as the document is represented as the sum of bag-

		en→de	de→en	it→de	it→es	en→es	Avg
Baselines	MultiCluster	92.9	69.1	79.1	81.0	63.1	77.0
	MultiCCA	69.2	50.7	83.1	79.0	45.3	65.5
	MultiSkip	79.9	63.5	71.8	76.3	60.4	70.4
	MultiTrans	87.7	75.2	70.4	64.4	56.1	70.8
Ours	Linear	83.8	75.7	74.8	67.3	57.4	71.8
	Joint	86.2	75.7	82.3	70.7	56.0	74.2
	Joint + Mapping	89.5	81.6	84.3	74.1	53.9	76.7
Bilingual	Luong et al. (2015)	88.4	80.3	-	-	-	-
	Chandar A P et al. (2014)	91.8	74.2	-	-	-	-
	Duong et al. (2016)	86.3	76.8	-	-	53.8	-

Table 3: Crosslingual document classification accuracy for various model. Chandar A P et al. (2014) and Luong et al. (2015) achieved a state-of-the-art result for $en \rightarrow de$ and $de \rightarrow en$ respectively, served as the reference. The best results for bilingual and multilingual word embeddings are bold.

of-word embeddings weighted by $tf.idf$. This setting is useful for target low-resource languages where the annotated data is insufficient.

The train and test data are from multilingual RCV1/RCV2 corpus (Lewis et al., 2004) where each document is annotated with labels from 4 categories: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social) and MCAT (Markets). We extend the evaluation from Klementiev et al. (2012) to cover more language pairs. We use the same data split for $en \rightarrow de$ and $de \rightarrow en$ pairs but additionally construct the train and test data for $it \rightarrow de$, $it \rightarrow es$ and $en \rightarrow es$. For each pair, we use 1,000 documents in the source language as the training data and 5,000 documents in the target language as the test data. The training data is randomly sampled, but the test data (for es) is evenly balanced among labels.

Table 3 shows the accuracy for the CLDC task for many pairs and models with respect to the baselines. For all bilingual models (Duong et al., 2016; Luong et al., 2015; Chandar A P et al., 2014), the bilingual word embeddings are constructed for each pair separately. In this way, they can only get the pairs involving English since there are many bilingual resources involving English on one side. For all our models, including Linear, Joint and Joint + Mapping, the embedding space is available for multiple languages; this is why we can exploit different relations, such as $it \rightarrow es$. This is the motivation for the work reported in this paper. Suppose we want to build a document clas-

sifier for es but lack any annotations. It is common to build $en-es$ crosslingual word embeddings for transfer learning, but this only achieves 53.8 % accuracy. Yet when we use it as the source, we get 81.0% accuracy. This is motivated by the fact that it and es are very similar.

The trend observed in Table 3 is consistent with previous observations. Linear transformation performs well. Joint training performs better especially for the $it \rightarrow de$ pair. The joint model with explicit mapping is generally our best model, even better than the base bilingual model from Duong et al. (2016). The $de \rightarrow en$ result improves on the existing state of the art reported in Luong et al. (2015). Our final model (Joint + Mapping) achieved competitive results compared with four strong baseline multilingual word embeddings, achieving best results for two out of five pairs. Moreover, the best scores for each language pairs are all from multilingual training, emphasizing the advantages over bilingual training.

9 Analysis

Mikolov et al. (2013b) showed that monolingual word embeddings capture some analogy relations such as $\vec{Paris} - \vec{France} + \vec{Italy} \approx \vec{Rome}$. It seems that in our multilingual embeddings, these relations still hold. Table 4 shows some examples of such relations where each word in the analogy query is in different languages.

All our baselines (MultiCluster, MultiCCA, MultiSkip, MultiTrans) are trained using different datasets. While MultiSkip and MultiTrans

chico _{es} - bruder _{de} + sorella _{it} (boy - brother + sister)	ehemann _{de} - padre _{es} + madre _{it} (husband - father + mother)	principe _{it} - junge _{de} + meisje _{nl} (prince - boy + girl)
chica _{es} (girl)	echtgenote _{nl} (wife)	principessa _{it} (princess)
ragazza _{it} (girl)	moglie _{it} (wife)	princess _{en}
meisje _{nl} (girl)	her _{en}	princesa _{es} (princess)
girl _{en}	marito _{it} (husband)	príncipe _{es} (prince)
mädchen _{de} (girl)	haar _{nl} (her)	prinzessin _{de} (princess)

Table 4: Top five closest words in our embeddings for multilingual word analogy. The transliteration is provided in parentheses. The correct output is bold.

	Tasks	MultiCluster	MultiCCA	Our model
Extrinsic	multilingual Dependency Parsing	61.0	58.7	61.2
	multilingual Document Classification	92.1	92.1	90.8
Intrinsic	monolingual word similarity	38.0	43.0	40.9
	multilingual word similarity	58.1	66.6	69.8
	word translation	43.7	35.7	45.7
	monolingual QVEC	10.3	10.7	11.9
	multilingual QVEC	9.3	8.7	8.6
	monolingual QVEC-CCA	62.4	63.4	46.4
	multilingual QVEC-CCA	43.3	41.5	31.0

Table 5: Performance of our model compared with MultiCluster and MultiCCA using extrinsic and intrinsic evaluation tasks on 12 languages proposed in Ammar et al. (2016), all models are trained on the same dataset. The best score for each task is bold.

are trained on parallel corpora, MultiCluster and MultiCCA use monolingual corpora and bilingual lexicons which are similar to our proposed methods. Therefore, for a strict comparison⁸, we train our best model (Joint + Mapping) using the same monolingual data and set of bilingual lexicons on the same 12 languages with MultiCluster and MultiCCA. Table 5 shows the performance on intrinsic and extrinsic tasks proposed in Ammar et al. (2016). Multilingual dependency parsing and document classification are trained on a set of source languages and test on a target language in the transfer learning setting. Monolingual word similarity task is similar with our monolingual similarity task described in §7, multilingual word similarity is an extension of monolingual word similarity task but tested for pair of words in different languages. Monolingual QVEC, multilingual QVEC test the linguistic content of word embeddings in monolingual and multilingual setting. Monolingual QVEC-CCA and multilingual QVEC-CCA are the

⁸also with respect to the word coverage since MultiSkip and MultiTrans usually have much lower word coverage, biasing the intrinsic evaluations.

extended versions of monolingual QVEC and multilingual QVEC also proposed in Ammar et al. (2016). Table 5 shows that our model achieved competitive results, best at 4 out of 9 evaluation tasks.

10 Conclusion

In this paper, we introduced several methods for building unified multilingual word embeddings. These represent an improvement because they exploit more relations and combine information from many languages. The input to our model is just a set of monolingual data and a set of bilingual lexicons between any language pairs. We induce the bilingual relationship for all language pairs while keeping high quality monolingual relations. Our multilingual joint training model with explicit mapping consistently achieves better performance compared with linear transformation. We achieve new state-of-the-art performance on bilingual lexicon induction task for recall at 5, similar excellent results with the state-of-the-art bilingual word embeddings on monolingual similarity task (Duong

et al., 2016). Moreover, our model is competitive at the crosslingual document classification task, achieving a new state of the art for $de \rightarrow en$ and $it \rightarrow de$ pair.

Acknowledgments

This work was conducted during Duong’s internship at IBM Research Tokyo and partially supported by the University of Melbourne and National ICT Australia (NICTA). We are grateful for support from NSF Award 1464553 and the DARPA/I2O, Contract Nos. HR0011-15-C-0114 and HR0011-15-C-0115. We thank Yuta Tsuboi and Alvin Grissom II for helpful discussions, and Doris Hoogeveen for helping with the `nl-es` evaluation.

References

- Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.
- Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.
- Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.
- Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning, ICML ’08*, pages 160–167, New York, NY, USA. ACM.
- Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Transgram, fast cross-lingual word-embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1113, Lisbon, Portugal, September. Association for Computational Linguistics.
- Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 600–609.
- Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015a. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122, Beijing, China, July. Association for Computational Linguistics.
- Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015b. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 845–850, Beijing, China. Association for Computational Linguistics.
- Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning Crosslingual Word Embeddings without Bilingual Corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, Austin, Texas, USA, November. Association for Computational Linguistics.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.
- Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web, WWW ’01*, pages 406–414, New York, NY, USA. ACM.

- Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 748–756. JMLR Workshop and Conference Proceedings.
- Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI’16, pages 2734–2740. AAAI Press.
- Kejun Huang, Matt Gardner, Evangelos Papalexakis, Christos Faloutsos, Nikos Sidiropoulos, Tom Mitchell, Partha P. Talukdar, and Xiao Fu. 2015. Translation invariant word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1084–1088, Lisbon, Portugal, September. Association for Computational Linguistics.
- David Kamholz, Jonathan Pool, and Susan Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, pages 3145–50, Reykjavik, Iceland. European Language Resources Association (ELRA).
- Alexandre Klementiev, Ivan Titov, and Binod Bhatnagar. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India, December. The COLING 2012 Organizing Committee.
- Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations with marginalizing alignments. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 224–229, Baltimore, Maryland, June. Association for Computational Linguistics.
- David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December.
- Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113.
- Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States.
- Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP ’11*, pages 62–72.
- Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1713–1722, Beijing, China, July. Association for Computational Linguistics.
- Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT ’12*, pages 477–487. Association for Computational Linguistics.
- Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725, Beijing, China, July. Association for Computational Linguistics.
- Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1082, Sofia, Bulgaria, August. Association for Computational Linguistics.

Min Xiao and Yuhong Guo, 2014. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, chapter Distributed Word Representation Learning for Cross-Lingual Dependency Parsing, pages 119–129. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8, Pittsburgh, Pennsylvania.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June. Association for Computational Linguistics.

Building Lexical Vector Representations from Concept Definitions

Danilo S. Carvalho and Minh Le Nguyen

Japan Advanced Institute of Science and Technology

1-1 Asahidai, Nomi City, Ishikawa, Japan

{danilo, nguyenml}@jaist.ac.jp

Abstract

The use of distributional language representations have opened new paths in solving a variety of NLP problems. However, alternative approaches can take advantage of information unavailable through pure statistical means. This paper presents a method for building vector representations from meaning unit blocks called concept definitions, which are obtained by extracting information from a curated linguistic resource (Wiktionary). The representations obtained in this way can be compared through conventional cosine similarity and are also interpretable by humans. Evaluation was conducted in semantic similarity and relatedness test sets, with results indicating a performance comparable to other methods based on single linguistic resource extraction. The results also indicate noticeable performance gains when combining distributional similarity scores with the ones obtained using this approach. Additionally, a discussion on the proposed method's shortcomings is provided in the analysis of error cases.

1 Introduction

Vector-based language representation schemes have gained large popularity in Natural Language Processing (NLP) research in the recent years. Their success comes from both the asserted benefits in several NLP tasks and from the ability to build them from unannotated textual data, widely available in the World Wide Web. The tasks benefiting from vector representations include Part-of-Speech (POS) tagging (dos Santos and Zadrozny, 2014), dependency parsing (Bansal et al., 2014), Named Entity Recognition (NER) (Seok et al.,

2016), Machine Translation (Sutskever et al., 2014), among others.

Such representation schemes are, however, not an all-in-one solution for the many NLP application scenarios. Thus, different representation methods were developed, each one focusing in a limited set of concerns, e.g., semantic relatedness measurement (Mikolov et al., 2013; Pennington et al., 2014) and grammatical dependencies (Levy and Goldberg, 2014). Most of the popular methods are based on a *distributional* approach: the meaning of a word is defined by the context of its use, i.e., the neighboring words. However, distributional representations carry no explicit linguistic information and cannot easily represent some important semantic relationships, such as synonymy and antonymy (Nguyen et al., 2016). Further problems include the difficulty in obtaining representations for out-of-vocabulary (OOV) words and complex constructs (collocations, idiomatic expressions), the lack of interpretable representations (Faruqui and Dyer, 2015), and the necessity of specific model construction for cross-language representation.

This paper presents a linguistically motivated language representation method, aimed at capturing and providing information unavailable on distributional approaches. Our contributions are: (i) a technique for building conceptual representations of linguistic elements (morphemes, words, collocations, idiomatic expressions) from a single collaborative language resource (*Wiktionary*¹); (ii) a method of combining said representations and comparing them to obtain a semantic similarity measurement. The conceptual representations, called *Term Definition Vectors*, address more specifically the issues of semantic relationship analysis, out-of-vocabulary word interpreta-

¹www.wiktionary.org

tion and cross-language conceptual mapping. Additionally, they have the advantages of being interpretable by humans and easy to operate, due to their sparsity. Experiments were conducted with the *SimLex-999* (Hill et al., 2015) test collection for word similarity, indicating a good performance in this task and exceeding the performance of other single information source studies, when combined with a distributional representation and Machine Learning. Error analysis was also conducted to understand the strengths and weaknesses of the proposed method.

The remainder of this paper is organized as follows: Section 2 presents relevant related works and highlights their similarities and differences to this research. Section 3 explains our approach in detail, covering its linguistic motivation and the characteristics of both representation model and comparison method. Section 4 describes the experimental evaluation and discusses the evaluation results and error analysis. Finally, Section 5 offers a summary of the findings and some concluding remarks.

2 Related Work

In order to address the limitations of the most popular representation schemes, approaches for all-in-one representation models were also developed (Pilehvar and Navigli, 2015; Derrac and Schockaert, 2015). They comprise a combination of techniques applied over different data sources for different tasks. Pilehvar and Navigli (2015) presented a method for combining Wiktionary and Wordnet (Fellbaum and others, 1998) sense information into a semantic network and a corresponding relatedness similarity measurement. The method is called ADW (Align, Disambiguate, Walk), and works by first using a Personalized PageRank (PPR) (Haveliwala, 2002) algorithm for performing a random walk on the semantic network and compute a *semantic signature* of a linguistic item (sense, word or text): a probability distribution over all entities in the network where the weights are estimated on the basis of the network's structural properties. Two linguistic items are then aligned and disambiguated by finding their two closest senses, comparing their semantic signatures under a set of vector and rank-based similarity measures (JensenShannon divergence, cosine, Rank-Biased Overlap, and Weighted Overlap). ADW achieved state-of-the-art performance

in several semantic relatedness test sets, covering words, senses and entire texts.

Recski et al. (2016) presented a hybrid system for measuring the semantic similarity of word pairs, using a combination of four distributional representations (*SENNa* (Collobert and Weston, 2008), (Huang et al., 2012), *word2vec* (Mikolov et al., 2013), and *GloVe* (Pennington et al., 2014)), *WordNet*-based features and *Alang* (Kornai, 2010) graph-based features to train a RBF kernel Support Vector Regression on the *SimLex-999* (Hill et al., 2015) data set. This system achieved state-of-the-art performance in *SimLex-999*.

The work presented in this paper takes a similar approach to Pilehvar and Navigli (2015), but stops short on obtaining a far reaching concept graph. Instead, it focuses on exploring the details of each sense definition. This includes term etymologies, morphological decomposition and translation links, available in Wiktionary. Another difference is that the translation links are used to map senses between languages in this work, whereas they are used for bridging gaps between sense sets on monolingual text in Pilehvar and Navigli (2015).

Another concern regarding distributional representations is their lack of interpretability from a linguistic standpoint. Faruqui and Dyer (2015) addresses this point, relying on linguistic information from Wordnet, Framenet (F. Baker et al., 1998), among other sources (excluding Wiktionary), to build interpretable word vectors. Such vectors accommodate several types of information, ranging from Part-of-Speech (POS) tags to sentiment classification and polarity. The obtained linguistic vectors achieved very good performance in a semantic similarity test. Those vectors, however, do not include morphological and translation information, offering discrete, binary features.

Regarding the extraction of definition data from Wiktionary, an effective approach is presented by Zesch et al. (2008a), which is also used for building a semantic representation (Zesch et al., 2008b). However, the level of detail and structure format obtained by such method was not deemed adequate for this work and an alternative extraction method was developed (Sections 3.2 and 3.3).

3 Term Definition Vectors

The basic motivation for the representation model here described is both linguistic and epistemic:

trying to represent knowledge as a set of individual concepts that relate to one another and are related to a set of terms. This idea is closely related to the Ogden/Richards *triangle of reference* (Ogden et al., 1923) (Figure 1), which describes a relationship between linguistic symbols and the objects they represent. The following notions are then defined:

- *Concept*: The unit of knowledge. Represents an individual meaning, e.g., rain (as in the natural phenomenon), and can be encoded into a term (symbol). It corresponds to the “thought or reference” from the triangle of reference.
- *Term*: A unit of perception. In text, it can be mapped to fragments ranging from morphemes to phrases. Each one can be decoded into one or more *concepts*. Stands for the “symbol” in the triangle of reference.
- *Definition*: A minimal, but complete explicitation of a concept. It comprises the textual explanation of the concept (sense) and its links to other concepts in a knowledge base, corresponding to the “symbolizes” relationship in the triangle of reference. The simplest case is a dictionary definition, consisting solely of a short explanation (typically a single sentence), with optional term highlights, linking to other dictionary entries. The information used for building definitions in this work is described in Section 3.3.

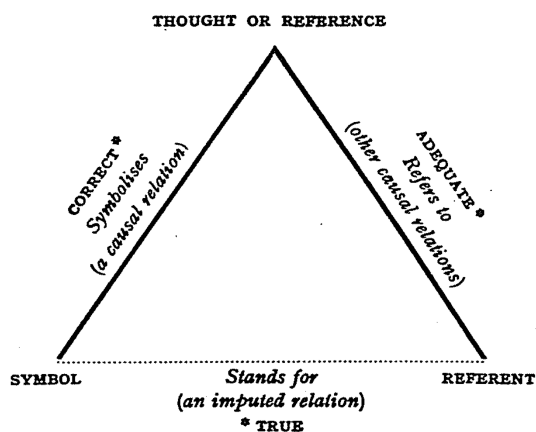


Figure 1: Ogden/Richards triangle of reference, also known as *semiotic triangle*. Describes a relationship between linguistic symbols and the objects they represent. (Ogden et al., 1923)

3.1 Distributional & Definitional Semantics

Distributional approaches for language representation, also known as *latent* or *statistical semantics*, are rooted in what is called the *distributional hypothesis* (Sahlgren, 2008). This concept stems from the notion that words are always used in a context, and it is the context that defines their meaning. Thus, the meaning of a term is concealed, i.e. latent, and can be revealed by looking at its context. In this sense, it is possible to define the meaning of a term to be a function of its neighboring term frequencies (co-occurrence). Using different definitions for “neighbor”, e.g., adjacent words in *word2vec* (Mikolov et al., 2013) and “modifiers in a dependency tree” (Levy and Goldberg, 2014), it is possible to produce a variety of vector spaces, called *embeddings*. Good embeddings enable the use of vector operations on words, such as comparison by cosine similarity. They also solve the data sparsity problem of large vocabularies, working as a dimensionality reduction method. There are, however, semantic elements that are not directly related to context, and thus are not well represented by distributional methods, e.g., the antonymy and hypernymy relations. Furthermore, polysemy can bring potential ambiguity problems in cases where the vectors are only indexed by surface form (word \rightarrow embedding).

An alternative line of thinking is to define the meanings first and then associate the corresponding terms (reference \rightarrow symbol). In this notion, meanings are *explicit* and need only to be resolved, i.e., disambiguated, for any given term. Concepts are thus represented by prior definitions instead of distributions over corpora, hence the name “*definitional semantics*” is used in this work to generalize such approaches.

To illustrate the difference between both approaches, a simple analogy can be made, where a person reads a book with difficult or new vocabulary. The distributional approach would be akin to reading the book while trying to guess the meaning of the unknown words by context. If the book is long, as the reading progresses, the guesses tend to become more accurate, as a human will try to piece together the information patterns surrounding the new words. On the other hand, the definitional approach would be equivalent to reading the entire contents of a dictionary before reading the book. The main advantage of the former is in-

dependence from any previously compiled knowledge base, e.g. a dictionary, which are subject to completeness and correctness concerns. The latter offers answers for the rarer words that are difficult to guess and the possibility to explain exactly how a certain meaning was inferred (interpretability).

The proposed definitional representation is then obtained through the following strategy:

1. Formalization of the basic unit of knowledge: the concept.
2. Information extraction from a linguistic resource into a set of concepts.
3. Lexical association: term \leftrightarrow concept.
4. Definition of a term as a composition (mixture) of concepts, allowing partial or complete disambiguation.

Figure 2 illustrates the process. A term is said ambiguous if it is composed by more than one concept. Therefore in this context, disambiguation is the action of reducing the number of concepts in a term's composition. This can be done by collecting additional information about the term, such as Part-of-Speech. A complete disambiguation reduces the composition to a single concept.

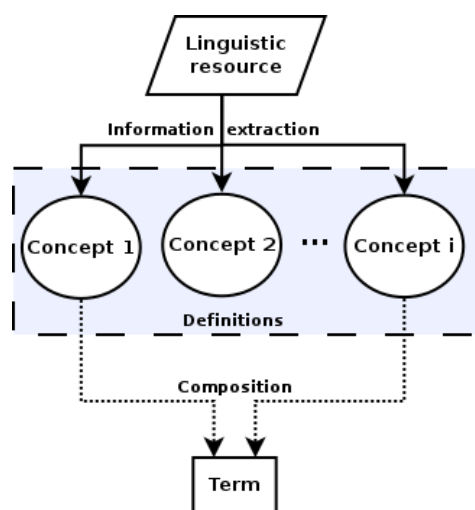


Figure 2: Process of definitional representation. Given a set of concepts obtained from a linguistic resource, a term can be defined as a composition of concepts. A term is said ambiguous if it is composed by more than one concept.

3.2 Linguistic Information Extraction

*Wiktionary*² was used as the single linguistic resource. Wiktionary is a collaborative lexical resource, comprising millions of vocabulary entries from several languages. It includes contextual information, etymology, semantic relations, translations, inflections, among other types of information for each entry. Its contents are actively curated by a large, global community. This choice was motivated by a several reasons, more importantly:

- It is the largest lexical resource openly available for the public, covering more than 10 million lexical entries from 172 languages.
- It is constantly updated. Daily changes are consolidated in monthly releases.
- Entries are organized in a way that separates each meaning of a term, simplifying definition extraction.
- Entries include range from morphemes, e.g., “pre-”, to idiomatic expressions, e.g., “take matters into one’s own hands”.

The data available from Wiktionary is semi-structured, composed of a set of markup documents, one for each entry, following a reasonably consistent standard of annotations for each language covered. In order to extract the linguistic information, an application was developed to convert the markup into a structured (JSON + schema) representation. The structured data was optimized for the retrieval of Wiktionary senses and link types were categorized to produce concept definitions.

3.3 Concept Definitions

Formalization of the knowledge unit used in this work was done by firstly mapping each concept to a single Wiktionary sense. The concept is represented as a lexical/semantic graph, where a main addressing term, the root node, is connected to other terms through a set of edges. Each edge denotes a different type of lexical/semantic relationship, e.g. prefixation, synonymy/antonymy. The edges are also weighted, denoting the intensity of a relationship. Figure 3 shows a simplified visualization of a pair of different concept graphs for

²www.wiktionary.org

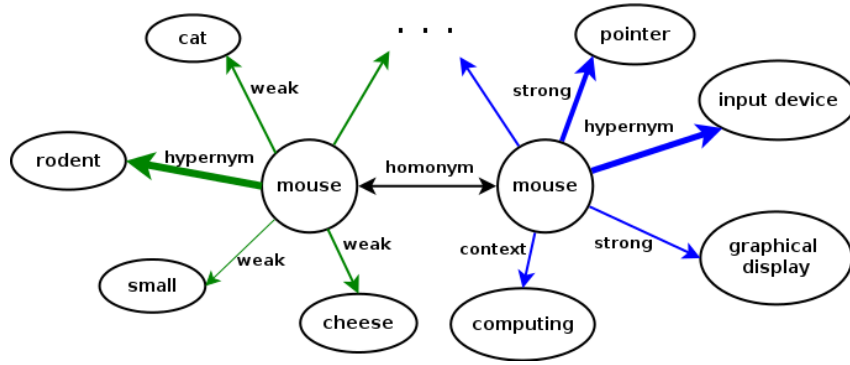


Figure 3: A simplified visualization of two concept graphs for the term “mouse”. The leftmost one denotes the concept of the small rodent and the other denotes the computer input device. The edge labels represent the relationship type and the thickness represent the its intensity.

Table 1: Link types used for the construction of concept graphs. They comprise both lexical (morphology, etymology) and semantic relationships between the root term, i.e., the Wiktionary entry title, and the terms used to describe the meaning.

Type	Description
weak	A term included in the description of the meaning on the Wiktionary entry.
strong	A term linked to another entry, i.e. a {highlight}, included in the description of the meaning.
context	A Wiktionary context link, explaining a specific situation in which the meaning described occurs.
synonym	A synonym relation. If it is an antonym, the sign of the link is reversed.
hypernym	A hypernym relation.
homonym	A homonym relation.
abbreviation	If the meaning described is given by interpreting the root term as an abbreviation.
etymology	Used to describe the origin of the root term of this meaning.
prefix	Denotes a prefixation (morphological) relationship of the root term.
suffix	Denotes a suffixation relationship. Same as above.
confix	Denotes a confixing relationship. Same as above.
affix	Denotes an affixation relationship. Same as above.
stem	Denotes a morphological stem relationship of the root term.
inflection	Denotes an inflectional relationship of the root term.

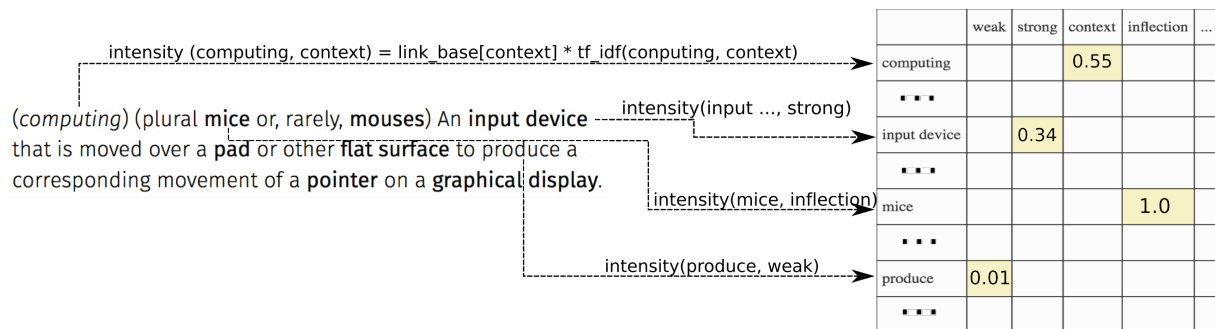


Figure 4: Representation of one Wiktionary sense definition for ”mouse” as an encoded matrix: the *concept definition*. Each Wiktionary link is categorized and mapped to a vector space.

a single lexical entry of Wiktionary. The different link types are used to create a vector space, in which the edges of the definition graph are represented. Table 1 describes the link types used in this work.

Each concept graph is represented by a $M \times T$ matrix called *concept definition*, where L is the

number of link types and T is the vocabulary size. The link intensities are defined for each type, by multiplying a manually defined constant *link_base* (a model parameter) by the TF-IDF score calculated for the vocabulary with respect to the type. Figure 4 illustrates the process.

Wiktionary entries also cover foreign terms,

listing senses in the source language, e.g., English meanings of the French word “avec” in the English language section. Definitions for these terms are also included into the concept definition set. Additionally, translation links are provided for many sense definitions. Such links, as well as term redirections, i.e., distinct terms pointing to the same Wiktionary entry, are mapped to a single concept. This allows foreign terms to take advantage of the same concept graphs as the source language equivalents.

3.4 Definition Vectors

Finally, association between the concept definitions and terms is established by composition. This is done by simple element-wise sum and average of all concept definition matrices mapped to a Wiktionary entry. The resulting matrix is flattened in its row axis, i.e, rows are concatenated in order, producing a $L \times T$ -dimensional sparse vector called *term definition vector*. If the term is not a Wiktionary entry, i.e., is out-of-vocabulary (OOV), a character n-gram-based attempt of morphological decomposition is done and if a complete morpheme match is found in the concept definition set, the matched concepts are composed for the OOV term. This decomposition attempt is done as follows:

1. For each character $c_i, i \in [0, n]$ in a OOV term of length n :
 - i Create empty list *morph_cand* of morpheme candidates.
 - ii Set index $j = i$.
 - iii Find Wiktionary entries that match c and add them to *morph_cand*. For the first and last characters, include prefixes and suffixes in the search, respectively.
 - iv Concatenate c_{j+1} to c .
 - v Increment j .
 - vi Repeat from iii.

This will produce a sequence of n morpheme candidate lists. If a sequence produced by taking a single morpheme candidate from each list matches the entire OOV term, it is considered a candidate decomposition. If there are multiple candidate decompositions, the one with the shortest stem is selected.

Figure 5 illustrates an OOV composition for the nonexistent term “unlockability”, which has

a complete morpheme match in the concept definition set. If a complete morpheme match is not found, the term is considered a proper noun (if no POS is provided), and given a null (zero) vector.

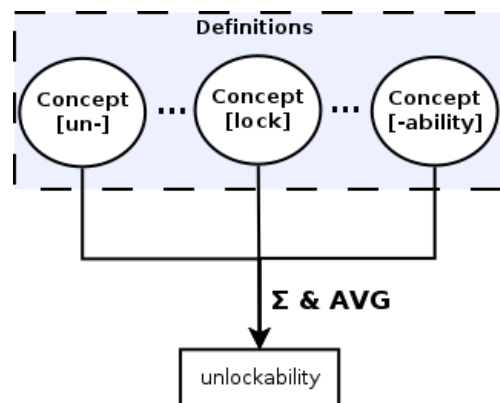


Figure 5: Morpheme match in concept definitions for the OOV term “unlockability”.

Similarity comparison between two terms is done by measuring the cosine similarity between their definition vectors. A value closer to 1 indicates high similarity, a value closer to -1 indicates opposition and a value closer to 0 indicates unrelatedness. Table 2 shows a comparison table between semantic matches obtained using this method, *word2vec* (Mikolov et al., 2013) and *GloVe* (Pennington et al., 2014).

Table 2: Top closest and farthest to the term “happy” by Term Definition Vector, and closest Word2Vec (GoogleNews corpus), and GloVe (Wikipedia2014 + Gigaword) cosine similarities.

Def.Vec	Def.Vec (-)	Word2Vec	GloVe
joyous	sad	glad	glad
dexterous	unhappy	pleased	good
content	joyless	ecstatic	sure
felicitous	somber	overjoyed	proud
lucky	depressed	thrilled	excited

The definition vectors obtained in this way are also human interpretable to a certain extent. Each dimension corresponds to a link from the concept graphs used to compose a term definition. The values correspond to the strengths of such links. A human readable representation of the definition vector for the word “sunny”, containing a maximum of two values per link type, can be written in the form: *weak@(a:0.0006, lot:0.003); strong@(cheerful:0.032, radiant:0.032); synonym@(bright:1.11, sunlit:1.11); suffix@(-y:1);*

stem@(sun:1); pos@(adjective:0.11, noun:0.11). In this example, *strong@radiant* is a single vector dimension and the term is not disambiguated (multiple POS).

4 Experiments

4.1 Experimental setup

The definition vector representations obtained in this work were evaluated in the *SimLex-999* test collection for semantic similarity benchmark (Hill et al., 2015). This test collection contains a set of 999 English word pairs, associated to a similarity score given by a group of human annotators. The set is divided in 666 nouns pairs, 222 verb pairs and 111 adjective pairs. The Part-of-Speech (POS) information allows partial or complete disambiguation of the definition vectors. The choice of *SimLex-999* was due to the type of similarity measured by this set, which excludes relatedness and is closer to the type of information captured by the concept definitions. Additionally, the *WordSim-353* (Finkelstein et al., 2001), *RG-65* (Rubenstein and Goodenough, 1965) and *MEN* (Bruni et al., 2014) test collections for semantic relatedness were also included in the evaluation, to verify the representation performance in measuring relatedness. While the *MEN* test collection also includes POS information, *WordSim-353* and *RG-65* do not include it, so sense distinction was not applied for the latter. Unfortunately, the test collections used in this work did not contain foreign words, so the translation-link features presented in Section 3.3 are solely presented as part of the method’s description, and are not evaluated. This is due to the method being developed without focus on a specific test. The semantic similarity measurement was consequence of the method’s design, but not its main target.

Evaluation is done by computing the Spearman’s rank correlation coefficient (ρ) between the human annotators’ similarity or relatedness scores and the scores given by the automated methods. A coefficient of value 1 means a perfect match between the relative positions of the pairs, when ranked by their similarity scores.

For the *SimLex-999* test, the cosine similarity between the term definition vectors was set as the similarity score. For the *WordSim-353*, *RG-65* and *MEN* tests, the absolute value of the cosine similarity was used instead, since opposite words are related. An additional test was performed to

explore the possibility of combining distributional and definitional approaches. In this test, a small set of features was created to train a Learning-to-Rank model, in order to improve the similarity scores. The features were as follows:

- Presence of synonym, hypernym, strong and weak links³ between the pair of words. Each link type is a separate feature.
- Term definition vector similarity.
- Word2vec similarity.

The features were computed for each pair and passed to SVM^{rank} (Joachims, 2006) for training and validation. A 10-fold cross-validation test using random pairs without replacement was run for the entire sets (5-fold for *RG-65*), except *MEN*. The *MEN* test collection is separated into training and testing sets, with 2K and 1K word pairs respectively, so these were used in place of the cross-validation. For each fold, the ranking scores provided by the trained ranker were used as similarity scores for calculating ρ . The average of all folds was considered the final result.

Experimental data and model parameters were set as follows:

- Linguistic information source: Wiktionary English database dump (XML + Wiki markup), 2015-12-26, containing more than 4 million entries. A reduced set, with only English, French, Greek, Japanese, Latin and Vietnamese language entries was used in the experiments. This set had about 734K entries, from which approx. 1 million concepts were extracted.
- *link_base* constants were set as: *weak* = 0.2, *strong* = 2.0, *context* = 0.5, *synonym* = 10.0, *hypernym* = 5.0, *homonym* = 7.0, *etymology* = 1.0 (also applied to morphological links) and *pos* = 1.0. The constants were adjusted by increasing or decreasing their values individually in intervals of 0.2, and observing the effect in ρ for *SimLex-999* in the first fold of the cross-validation. The optimal values were selected and kept constant for the remaining folds and for the other tests. This was done because

³See Table 1

Table 3: Performance of different methods for the SimLex-999, WordSim-353, RG-65, and MEN test sets, reported as Spearman’s rank correlation coefficient ρ . The methods marked with \diamond use a single information source. Fields marked with “-” indicate that the results were not available for assessment.

Method	ρ @SimLex-999	ρ @WordSim-353	ρ @RG-65	ρ @MEN-1K
Word2Vec (W2V) \diamond	0.38	0.78	0.84	0.73
GloVe \diamond	0.40	0.76	0.83	-
Term Def. Vectors (TDV) \diamond	0.56	0.36	0.68	0.42
Ling Dense	0.58	0.45	0.67	-
dLCE \diamond	0.59	-	-	-
TDV + W2V + SVM ^{rank}	0.62	0.75	0.72	0.78
Recski et al. (2016)	0.76	-	-	-
ADW	-	0.75	0.92	-

changing *link_base* for each fold would create an unrealistic use scenario for our system, which cannot change *link_base* online. The cross-validation was repeated two times, with very minor differences between both test runs. The constant values reported here are from the last run.

- SVM^{rank} was set with a default linear kernel and C parameter (training error trade-off) was set to 8 for MEN and 5 for the other test collections. The value was increased in unit intervals, until convergence was longer than a time threshold (10 minutes). This parameter was adjusted using the training set for MEN, or inside each CV fold for the rest.
- Both *Word2Vec* and *GloVe* were used with pre-trained, 300-dimensional models: 100 billion words GoogleNews corpus and Common Crawl 42 billion token corpus respectively.

dLCE (Nguyen et al., 2016) was chosen as baseline, for being the best single information source method in the SimLex-999 test collection. Further results include Recski et al. (2016) (state-of-the-art), Ling Dense (Faruqui and Dyer, 2015), Word2Vec (Mikolov et al., 2013), and GloVe (Pennington et al., 2014). For WordSim-353, GloVe, Word2Vec, Ling Dense, and ADW (Pilehvar and Navigli, 2015) were included. For RG-65, Ling Dense, GloVe, and ADW (state-of-the-art), were included.

4.2 Results

The experimental results are presented in Table 3, where they are compared to other methods.

The results indicate that in the semantic similarity test, the term definition vectors perform closely to other representation models taking advantage of curated data, such as WordNet. It also outperforms the most popular distributional representations. However, they are clearly outclassed in the semantic relatedness test, for which the distributional approaches show superior performance.

An interesting observation can be made when combining word2vec similarity with term definition features through the use of Machine Learning. A performance trade-off seems to exist at the semantic relatedness tests, but the same is not true for the similarity test. This allowed the combined model to improve considerably at little cost. Further analysis helped in understanding the reason for this particularity (Section 4.3).

Lastly, the experiments have also shown that the method for extracting concept definitions is not computationally expensive. The developed implementation took about 6 minutes to extract all concept definitions from the structured Wiktionary data used in the tests, using a modern desktop computer (3GHz processor and at least 8GB RAM). Structuring Wiktionary data took less than 20 minutes with the same equipment, and was done a single time.

4.3 Error analysis

Identifying the flaws in a method is a fundamental step in improving it and also in understanding the problem it tries to solve. With this in mind, the error cases identified in measuring similarity from the SimLex-999 set were observed in detail. In

this analysis we considered as error any word pair that was put among the top 15% similarity scores by the human annotators, but was ranked in the lower 50% using the definition vectors. The same applies for the bottom 15% scored by humans, that are ranked in the upper half by our approach.

The errors found were classified in four categories:

- **Insufficient links in Wiktionary:** this type of error occurs when the wiktionary sense corresponding to a concept lacks annotations. Typical cases contain only a short description, with no links. The concept graph is then left with only weak links, which have little impact on similarity calculation. The pair *drizzle–rain* (noun) is one example of this.
- **Undeclared hypernymy:** certain cases of hypernymy are not solved in the concept extraction, since they require multiple hops in the definition links to be found. The pairs *cop–sheriff* and *alcohol–gin* (noun) are instances of such problem.
- **Casual vs. formal language semantics:** not a flaw in the method per se, but an error caused by the differences in formal description of a language (in a dictionary), when compared to casual use. The pair *noticeable–obvious* (adjective) illustrates this.
- **Other:** flaws in the extraction process or annotation problems in Wiktionary.

Those errors affect the pairs in the top 15% human similarity scores 7 times more than the lower 15%. They are distributed as shown in Table 4.

Table 4: Distribution of definition vector error types in SimLex-999.

Type of error	Proportion
Insufficient links	21.4%
Undeclared hypernymy	38.1%
Casual semantics	14.3%
Other	26.2%

Having about one quarter of the errors in the “other” category shows that there is some space for improvement in the concept extraction process. The insufficient links and undeclared hypernymy categories are cases in which distributional approaches may do better if similarity is high, due to the words intrinsic relatedness.

Analysis of SVM^{rank} scores showed that the insufficient links category benefited the most from the combination with word2vec. The reason is that the features chosen for use with the ranker made such cases distinguishable and more likely to receive a larger weight from the word2vec similarity score after training. The undeclared hypernymy cases, on the other hand, are not so evident and would require a more complex approach on the concept extraction process.

5 Conclusion

Alternative approaches to distributional language representations can take advantage of information unavailable through pure statistical means. Taking advantage of large curated linguistic resources is a popular way of obtaining such information and offers large room for exploration. With this in mind, we propose a novel method for obtaining vector representations of lexical items using Wiktionary sense definitions. The lexical item representations are composed from basic meaning units called concept definitions, which are extracted from the linguistic resource. Results obtained from a semantic similarity evaluation test indicate performance comparable to other methods based on linguistic resource extraction. Furthermore, a noticeable performance gain was obtained by applying a Machine Learning approach to combine word2vec similarity scores with the ones obtained using this approach, exceeding both methods’ results.

Planned improvements include the use of a graph traversal approach to capture deeper semantic links and also the inclusion of translation links as separate dimensions in the concept vector space, in order to facilitate the use of obtained representations as a machine translation resource. The inclusion of distributional similarity measures as separate dimensions is also under study and provides an alternative way of combining the distributional and definitional approaches.

Acknowledgments

We also would like to thank anonymous reviewers for their detailed comments for improving our paper. This work was partially supported by CNPq (National Counsel of Technological and Scientific Development) - Brazil, JSPS KAKENHI Grant number 15K16048, JSPS KAKENHI Grant Number JP15K12094, and CREST, JST.

References

- Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics.
- Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*, 49(1-47).
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- Joaquín Derrac and Steven Schockaert. 2015. Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning. *Artificial Intelligence*, 228:66–94.
- Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.
- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.
- Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 464–469. Association for Computational Linguistics.
- Christiane Fellbaum et al. 1998. Wordnet: An electronic database.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppín. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.
- Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM.
- Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.
- Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882. Association for Computational Linguistics.
- Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.
- András Kornai. 2010. The algebra of lexical semantics. In *The Mathematics of Language*, pages 174–199. Springer.
- Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308. Association for Computational Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.
- Anh Kim Nguyen, Sabine Schulte im Walde, and Thang Ngoc Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459. Association for Computational Linguistics.
- Charles Kay Ogden, Ivor Armstrong Richards, Sv Ransulf, and E Cassirer. 1923. The meaning of meaning. a study of the influence of language upon thought and of the science of symbolism.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.
- Mohammad Taher Pilehvar and Roberto Navigli. 2015. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128.
- Gábor Recski, Eszter Iklódi, Katalin Pajkossy, and Andras Kornai, 2016. *Proceedings of the 1st Workshop on Representation Learning for NLP*, chapter Measuring Semantic Similarity of Words Using Concept Networks, pages 193–200. Association for Computational Linguistics.
- Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.
- Magnus Sahlgren. 2008. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54.
- Miran Seok, Hye-Jeong Song, Chan-Young Park, Jong-Dae Kim, and Yu-seop Kim. 2016. Named entity recognition using word embedding as a feature. *International Journal of Software Engineering and Its Applications*, 10(2):93–104.

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

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008a. Extracting lexical semantic knowledge from wikipedia and wiktionary. In *LREC*, volume 8, pages 1646–1652.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008b. Using wiktionary for computing semantic relatedness. In *AAAI*, volume 8, pages 861–866.

ShotgunWSD: An unsupervised algorithm for global word sense disambiguation inspired by DNA sequencing

Andrei M. Butnaru, Radu Tudor Ionescu and Florentina Hristea

University of Bucharest
Department of Computer Science
14 Academiei, Bucharest, Romania
butnaruandreimadalin@gmail.com
raducu.ionescu@gmail.com
fhristea@fmi.unibuc.ro

Abstract

In this paper, we present a novel unsupervised algorithm for word sense disambiguation (WSD) at the document level. Our algorithm is inspired by a widely-used approach in the field of genetics for whole genome sequencing, known as the Shotgun sequencing technique. The proposed WSD algorithm is based on three main steps. First, a brute-force WSD algorithm is applied to short context windows (up to 10 words) selected from the document in order to generate a short list of likely sense configurations for each window. In the second step, these local sense configurations are assembled into longer composite configurations based on suffix and prefix matching. The resulted configurations are ranked by their length, and the sense of each word is chosen based on a voting scheme that considers only the top k configurations in which the word appears. We compare our algorithm with other state-of-the-art unsupervised WSD algorithms and demonstrate better performance, sometimes by a very large margin. We also show that our algorithm can yield better performance than the Most Common Sense (MCS) baseline on one data set. Moreover, our algorithm has a very small number of parameters, is robust to parameter tuning, and, unlike other bio-inspired methods, it gives a deterministic solution (it does not involve random choices).

1 Introduction

Word Sense Disambiguation (WSD), the task of identifying which sense of a word is used in a

given context, is a core NLP problem, having the potential to improve many applications such as machine translation (Carpuat and Wu, 2007), text summarization (Plaza et al., 2011), information retrieval (Chifu and Ionescu, 2012; Chifu et al., 2014) or sentiment analysis (Sumanth and Inkpen, 2015). Most of the existing WSD algorithms (Agirre and Edmonds, 2006; Navigli, 2009) are commonly classified into supervised, unsupervised, and knowledge-based techniques, but hybrid approaches have also been proposed in the literature (Hristea et al., 2008). The main disadvantage of supervised methods (that have led to the best disambiguation results) is that they require a large amount of annotated data which is difficult to obtain. Hence, over the last few years, many researchers have concentrated on developing unsupervised learning approaches (Schwab et al., 2012; Schwab et al., 2013a; Schwab et al., 2013b; Chen et al., 2014; Bhingardive et al., 2015). In this paper, we introduce a novel WSD algorithm, termed ShotgunWSD¹, that stems from the Shotgun genome sequencing technique (Anderson, 1981; Istrail et al., 2004). Our WSD algorithm is also unsupervised, but it requires knowledge in the form of WordNet (Miller, 1995; Fellbaum, 1998) synsets and relations as well. Thus, our algorithm can be regarded as a hybrid approach.

WSD algorithms can perform WSD at the local or at the global level. A local WSD algorithm, such as the extended Lesk measure (Lesk, 1986; Banerjee and Pedersen, 2002; Banerjee and Pedersen, 2003), is designed to assign the appropriate sense, from an existing sense inventory, for a target word in a given context window of a few words. For instance, for the word “sense” in the context “You have a good sense of humor.”, the

¹Our open source Java implementation of ShotgunWSD is freely available at <http://ai.fmi.unibuc.ro/Home/Software>.

sense that corresponds to the *natural ability* rather than the *meaning of a word* or the *sensation* should be chosen by a WSD algorithm. Rather more generally, a global WSD approach aims to choose the appropriate sense for each ambiguous word in a text document. The straightforward solution is the exhaustive evaluation of all sense combinations (configurations) (Patwardhan et al., 2003), but the time complexity is exponential with respect to the number of words in the text, as also noted by Schwab et al. (2012), Schwab et al. (2013a). Indeed, the brute-force (BF) solution quickly becomes impractical for windows of more than a few words. Hence, several approximation methods have been proposed for the global WSD task in order to overcome the exponential growth of the search space (Schwab et al., 2012; Schwab et al., 2013a). Our algorithm is designed to perform global WSD by combining multiple local sense configurations that are obtained using BF search, thus avoiding BF search on the whole text. A local WSD algorithm is employed to build the local sense configurations. We alternatively use two methods at this step, namely the extended Lesk measure (Banerjee and Pedersen, 2002; Banerjee and Pedersen, 2003) and an approach based on deriving sense embeddings from word embeddings (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013). Both local WSD approaches are based on WordNet synsets and relations.

Our global WSD algorithm can be briefly described in a few steps. In the first step, context windows of a fixed length n are selected from the document, and for each context window the top scoring sense configurations constructed by BF search are kept for the second step. The retained sense configurations are merged based on suffix and prefix matching. The configurations obtained so far are ranked by their length (the longer, the better), and the sense of each word is given by a majority vote on the top k configurations that cover the respective word. Compared to other state-of-the-art bio-inspired methods (Schwab et al., 2012; Schwab et al., 2013a), our algorithm has less parameters. Different from the other methods, these parameters (n and k) can be intuitively tuned with respect to the WSD task. As we select a single context window at every possible location in a text, our algorithm becomes deterministic, obtaining the same global configuration for a given set

of parameters and input document. Thus, our algorithm is not affected by random chance, unlike stochastic algorithms such as Ant Colony Optimization (Lafourcade and Guinand, 2010; Schwab et al., 2012; Schwab et al., 2013a).

We have conducted experiments on SemEval 2007 (Navigli et al., 2007), Senseval-2 (Edmonds and Cotton, 2001) and Senseval-3 (Mihalcea et al., 2004) data sets in order to compare ShotgunWSD with three state-of-the-art approaches (Schwab et al., 2013a; Chen et al., 2014; Bhingardive et al., 2015) along with the Most Common Sense (MCS) baseline², which is considered as the strongest baseline in WSD (Agirre and Edmonds, 2006). The empirical results show that our algorithm compares favorably to these approaches.

The rest of this paper is organized as follows. Related work on unsupervised WSD algorithms is presented in Section 2. The ShotgunWSD algorithm is described in Section 3. The experiments are given in Section 4. Finally, we draw our conclusions in Section 5.

2 Related Work

There is a broad range of methods designed to perform WSD (Agirre and Edmonds, 2006; Navigli, 2009; Vidhu Bhala and Abirami, 2014). The most accurate techniques are supervised (Iacobacci et al., 2016), but they require annotated training data which is not always available. In order to overcome this limitation, some researchers have proposed various unsupervised or knowledge-based WSD methods (Banerjee and Pedersen, 2002; Banerjee and Pedersen, 2003; Schwab et al., 2012; Nguyen and Ock, 2013; Schwab et al., 2013a; Schwab et al., 2013b; Chen et al., 2014; Agirre et al., 2014; Bhingardive et al., 2015). Since our approach is unsupervised and based on WordNet (Miller, 1995; Fellbaum, 1998), our focus is to present related work in the same direction. Banerjee and Pedersen (2002) extend the gloss overlap algorithm of Lesk (1986) by using WordNet relations. Patwardhan et al. (2003) proposed a brute-force algorithm for global WSD by employing the extended Lesk measure (Banerjee and Pedersen, 2002; Banerjee and Pedersen, 2003) to compute the semantic relatedness among senses in a given text. However, their BF approach is not suitable for whole text documents, because of the high computational time. More recently, Schwab

²Also known as the Most Frequent Sense baseline.

et al. (2012) have proposed and compared three stochastic algorithms for global WSD as alternatives to BF search, namely a Genetic Algorithm, Simulated Annealing, and Ant Colony Optimization. Among these, the authors (Schwab et al., 2012; Schwab et al., 2013a) have found that the Ant Colony Algorithm yields better results than the other two.

Recently, word embeddings have been used for WSD (Chen et al., 2014; Bhingardive et al., 2015; Iacobacci et al., 2016). Word embeddings are well known in the NLP community (Bengio et al., 2003; Collobert and Weston, 2008), but they have recently become more popular due to the work of Mikolov et al. (2013) which introduced the *word2vec* framework that allows to efficiently build vector representations from words. Chen et al. (2014) present a unified model for joint word sense representation and disambiguation. They use the Skip-gram model to learn word vectors. On the other hand, Bhingardive et al. (2015) use pre-trained word vectors to build sense embeddings by averaging the word vectors produced for each sense of a target word. As their goal is to find an approximation for the MCS baseline, they try to find the sense embedding that is closest to the embedding of the target word. Iacobacci et al. (2016) propose different methods through which word embeddings can be leveraged in a *supervised* WSD system architecture. Remarkably, they find that a WSD approach based on word embeddings alone can provide significant performance improvements over a state-of-the-art WSD system that uses standard WSD features.

3 ShotgunWSD

As also noted by Schwab et al. (2012), brute-force WSD algorithms based on semantic relatedness (Patwardhan et al., 2003) are not practical for whole text disambiguation due to their exponential time complexity. In this section, we describe a novel WSD algorithm that aims to avoid this computational issue. Our algorithm is inspired by the Shotgun genome sequencing technique (Anderson, 1981) which is used in genetics research to obtain long DNA strands. For example, Istrail et al. (2004) have used this technique to assemble the human genome. Before a long DNA strand can be read, Shotgun sequencing needs to create multiple copies of the respective strand. Next, DNA is randomly broken

down into many small segments called *reads* (usually between 30 and 400 nucleotides long), by adding a restriction enzyme into the chemical solution containing the DNA. The reads can then be sequenced using Next-Generation Sequencing technology (Voelkerding et al., 2009), for example by using an Illumina (Solexa) machine (Bennett, 2004). In genome assembly, the low quality reads are usually eliminated (Patel and Jain, 2012) and the whole genome is reconstructed by assembling the remaining reads. One strategy is to merge two or more reads in order to obtain longer DNA segments, if they have a significant overlap of matching nucleotides. Because of reading errors or mutations, the overlap is usually measured using a distance measure, e.g. edit distance (Levenshtein, 1966). If a backbone DNA sequence is available, the reads are aligned to the backbone DNA before assembly, in order to find their approximate position in the DNA that needs to be reconstructed.

We next present how we adapt the Shotgun sequencing technique for the task of global WSD. We will make a few observations along the way that will lead to a simplified method, namely *ShotgunWSD*, which is formally presented in Algorithm 1. We use the following notations in Algorithm 1. An array (or an ordered set of elements) is denoted by $X = (x_1, x_2, \dots, x_m)$ and the length of X is denoted by $|X| = m$. Arrays are considered to be indexed starting from position 1, thus $X[i] = x_i, \forall i \in \{1, 2, \dots, m\}$.

Our goal is to find a configuration of senses G for the whole document D , that matches the ground-truth configuration produced by human annotators. A *configuration of senses* is simply obtained by assigning a sense to each word in a text. In this work, the senses are selected from WordNet (Miller, 1995; Fellbaum, 1998), according to steps 7-8 of Algorithm 1. Naturally, we will consider that the sense configuration of the whole document corresponds to the long DNA strand that needs to be sequenced. In this context, sense configurations of short context windows (less than 10 words) will correspond to the short DNA reads. A crucial difference here is that we know the location of the context windows in the whole document from the very beginning, so our task will be much easier compared to Shotgun sequencing (we do not need to use a backbone solution for the alignment of short sense configurations). At every possible location in the text document (step

Algorithm 1: ShotgunWSD Algorithm

```
1 Input:
2  $D = (w_1, w_2, \dots, w_m)$  – a document of  $m$  words denoted by  $w_i, i \in \{1, 2, \dots, m\}$ ;
3  $n$  – the length of the context windows ( $1 < n < m$ );
4  $k$  – the number of sense configurations considered for the voting scheme ( $k > 0$ );

5 Initialization:
6  $c \leftarrow 20$ ;
7 for  $i \in \{1, 2, \dots, m\}$  do
8    $S_{w_i} \leftarrow$  the set of WordNet synsets of  $w_i$ ;
9  $S \leftarrow \emptyset$ ;
10  $G \leftarrow (0, 0, \dots, 0)$ , such that  $|G| = m$ ;

11 Computation:
12 for  $i \in \{1, 2, \dots, m - n + 1\}$  do
13    $C_i \leftarrow \emptyset$ ;
14   while did not generate all sense configurations do
15      $C \leftarrow$  a new configuration  $(s_{w_i}, s_{w_{i+1}}, \dots, s_{w_{i+n-1}}), s_{w_j} \in S_{w_j}, \forall j \in \{i, \dots, i + n - 1\}$ , such that  $C \notin C_i$ ;
16      $r \leftarrow 0$ ;
17     for  $p \in \{1, 2, \dots, n - 1\}$  do
18       for  $q \in \{p + 1, 2, \dots, n\}$  do
19          $r \leftarrow r + \text{relatedness}(C[p], C[q])$ ;
20      $C_i \leftarrow C_i \cup \{(C, i, n, r)\}$ ;
21    $C_i \leftarrow$  the top  $c$  configurations obtained by sorting the configurations in  $C_i$  by their relatedness score (descending);
22    $S \leftarrow S \cup C_i$ ;

23 for  $l \in \{\min\{4, n - 1\}, \dots, 1\}$  do
24   for  $p \in \{1, 2, \dots, |S|\}$  do
25      $(C_p, i_p, n_p, r_p) \leftarrow$  the  $p$ -th component of  $S$ ;
26     for  $q \in \{1, 2, \dots, |S|\}$  do
27        $(C_q, i_q, n_q, r_q) \leftarrow$  the  $q$ -th component of  $S$ ;
28       if  $i_q - i_p < n_p$  and  $i_p \neq i_q$  then
29          $t \leftarrow \text{true}$ ;
30         for  $x \in \{1, \dots, l\}$  do
31           if  $C_p[n_p - l + x] \neq C_q[x]$  then
32              $t \leftarrow \text{false}$ ;
33         if  $t = \text{true}$  then
34            $C_{p \oplus q} \leftarrow (C_p[1], C_p[2], \dots, C_p[n_p], C_q[l + 1], C_q[l + 2], \dots, C_q[n_q])$ ;
35            $r_{p \oplus q} \leftarrow r_p$ ;
36           for  $i \in \{1, 2, \dots, n_p + n_q - l\}$  do
37             for  $j \in \{l + 1, l + 2, \dots, n_q\}$  do
38                $r_{p \oplus q} \leftarrow r_{p \oplus q} + \text{relatedness}(C_{p \oplus q}[i], C_q[j])$ ;
39            $S \leftarrow S \cup \{(C_{p \oplus q}, i_p, n_p + n_q - l, r_{p \oplus q})\}$ ;

40 for  $j \in \{1, 2, \dots, m\}$  do
41    $Q_j \leftarrow \{(C, i, d, r) \mid (C, i, d, r) \in S, j \in \{i, i + 1, \dots, i + d - 1\}\}$ ;
42    $Q_j \leftarrow$  the top  $k$  configurations obtained by sorting the configurations in  $Q_j$  by their length (descending);
43    $ps_{w_j} \leftarrow$  the predominant sense obtained by using a majority voting scheme on  $Q_j$ ;
44    $G[j] \leftarrow ps_{w_j}$ ;

45 Output:
46  $G = (ps_{w_1}, ps_{w_2}, \dots, ps_{w_m}), ps_{w_i} \in S_{w_i}, \forall i \in \{1, 2, \dots, m\}$  – the global configuration of senses returned by the algorithm.
```

12), we select a window of n words. The window length n is an external parameter of our algorithm that can be tuned for optimal results. For each context window we will compute all possible sense configurations (steps 14-15). A score is assigned to each sense configuration by using the semantic relatedness between word senses (steps 16-19), as described by Patwardhan et al. (2003).

We alternatively employ two measures to compute the semantic relatedness, one is the extended Lesk measure (Banerjee and Pedersen, 2002; Banerjee and Pedersen, 2003) and the other is a simple approach based on deriving sense embeddings from word embeddings (Mikolov et al., 2013). Both methods are described in Section 3.1. We will keep the top scoring sense configurations (step 21)

for the assembly phase (steps 23-39). In step 21, we use an internal parameter c in order to determine exactly how many sense configurations are kept per context window. Another important remark is that we assume that the BF algorithm used to obtain sense configurations for short windows does not produce output errors, so it is not necessary to use a distance measure in order to find overlaps for merging configurations. We simply check if the suffix of a former configuration coincides with the prefix of a latter configuration in order to join them together (steps 29-33). The length l of the suffix and the prefix that get overlapped needs to be greater than zero, so at least one sense choice needs to coincide. We gradually consider shorter and shorter suffix and prefix lengths starting with $l = \min\{4, n - 1\}$ (step 23). Sense configurations are assembled in order to obtain longer configurations (step 34), until none of the resulted configurations can be further merged together. When merging, the relatedness score of the resulting configuration needs to be recomputed (steps 36-38), but we can take advantage of some of the previously computed scores (step 35). Longer configurations indicate that there is an agreement (regarding the chosen senses) that spans across a longer piece of text. In other words, longer configurations are more likely to provide correct sense choices, since they inherently embed a higher degree of agreement among senses. After the configuration assembly phase, we start assigning the sense to each word in the document (step 40). Based on the assumption that longer configurations provide better information, we build a ranked list of sense configurations for each word in the document (step 42). Naturally, for a given word, we only consider the configurations that contain the respective word (step 41). Finally, the sense of each word is given by a majority vote on the top k configurations from its ranked list (steps 43-44). The number of sense configurations k is an external parameter of our approach, and it can be tuned for optimal results.

3.1 Semantic Relatedness

For a sense configuration of n words, we compute the semantic relatedness between each pair of selected senses. We use two different approaches for computing the relatedness score and both of them are based on WordNet semantic relations. In this context, we essentially need to compute the se-

mantic relatedness of two WordNet synsets. For each synset we build a disambiguation vocabulary by extracting words from the WordNet lexical knowledge base, as follows. Starting from the synset itself, we first include all the synonyms that form the respective synset along with the content words of the gloss (examples included). We also include into the disambiguation vocabulary words indicated by specific WordNet semantic relations that depend on the part-of-speech of the word. More precisely, we have considered hyponyms and meronyms for nouns. For adjectives, we have considered similar synsets, antonyms, attributes, pertainyms and related (see also) synsets. For verbs, we have considered troponyms, hypernyms, entailments and outcomes. Finally, for adverbs, we have considered antonyms, pertainyms and topics. These choices have been made because previous studies (Banerjee and Pedersen, 2003; Hristea et al., 2008) have come to the conclusion that using these specific relations for each part-of-speech seems to provide useful information in the WSD process. The disambiguation vocabulary generated by the WordNet feature selection described so far needs to be further processed in order to obtain the final vocabulary. The first processing step is to eliminate the stopwords. The remaining words are stemmed using the Porter stemmer algorithm (Porter, 1980). The resulted stems represent the final set of features that we use to compute the relatedness score between two synsets. The two measures that we employ for computing the relatedness score are described next.

3.1.1 Extended Lesk Measure

The original Lesk algorithm (Lesk, 1986) only considers one word overlaps among the glosses of a target word and those that surround it in a given context. Banerjee and Pedersen (2002) note that this is a significant limitation because dictionary glosses tend to be fairly short and they fail to provide sufficient information to make fine grained distinctions required for WSD. Therefore, Banerjee and Pedersen (2003) introduce a measure that takes as input two WordNet synsets and returns a numeric value that quantifies their degree of semantic relatedness by taking into consideration the glosses of related WordNet synsets as well. Moreover, when comparing two glosses, the extended Lesk measure considers overlaps of multiple consecutive words, based on the assumption that the

longer the phrase, the more representative it is for the relatedness of the two synsets. Given two input glosses, the longest overlap between them is detected and then replaced with a unique marker in each of the two glosses. The resulted glosses are then again checked for overlaps, and this process continues until there are no more overlaps. The lengths of the detected overlaps are squared and added together to obtain the score for the given pair of glosses. Depending on the WordNet relations used for each part-of-speech, several pairs of glosses are compared and summed up together to obtain the final relatedness score. However, if the two words do not belong to the same part-of-speech, we only use their WordNet glosses and examples. Further details regarding this approach are provided by Banerjee and Pedersen (2003).

3.1.2 Sense Embeddings

A simple approach based on word embeddings is employed to measure the semantic relatedness of two synsets. *Word embeddings* (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013) represent each word as a low-dimensional real valued vector, such that related words reside in close vicinity in the generated space. We have used the pre-trained word embeddings computed by the *word2vec* toolkit (Mikolov et al., 2013) on the Google News data set using the Skip-gram model. The pre-trained model contains 300-dimensional vectors for 3 million words and phrases.

The relatedness score between two synsets is computed as follows. For each word in the disambiguation vocabulary that represents a synset, we compute its word embedding vector. Thus, we obtain a cluster of word embedding vectors for each given synset. Sense embeddings are then obtained by computing the centroid of each cluster as the median of all the word embeddings in the respective cluster. We can naturally assume that some of the words in the cluster may actually be outliers. Thus, we believe that using the (geometric) median instead of the mean is more appropriate, as the mean is largely influenced by outliers. Finally, the semantic relatedness of two synsets is simply given by the cosine similarity between their cluster centroids.

It is important to note that an approach based on the mean of word vectors to construct sense embeddings is used by Bhingardive et al. (2015), but with a slightly different purpose than ours, namely

to determine which synset better fits a target word, assuming that this synset should correspond to the most common sense of the respective word. As such, they completely disregard the context of the target word. Different from their approach, we are trying to find how related two synsets of distinct words that appear in the same context window are. Furthermore, the empirical results presented in Section 4 show that our approach yields better performance than the MCS estimation method of Bhingardive et al. (2015), thus putting a greater gap between the two methods.

4 Experiments and Results

4.1 Data Sets

We compare our global WSD algorithm with several state-of-the-art unsupervised WSD methods using the same test data as in the works presenting them.

We first compare ShotgunWSD with two state-of-the-art approaches (Schwab et al., 2013a; Chen et al., 2014) and the MCS baseline, on the SemEval 2007 coarse-grained English all-words task (Navigli et al., 2007). The SemEval 2007 coarse-grained English all-words data set³ is composed of 5 documents that contain 2269 ambiguous words (1108 nouns, 591 verbs, 362 adjectives, 208 adverbs) altogether. We also compare our approach with the MCS estimation method of Bhingardive et al. (2015), the MCS baseline and the extended Lesk algorithm (Torres and Gelbukh, 2009) on the Senseval-2 English all-words (Edmonds and Cotton, 2001) and the Senseval-3 English all-words (Mihalcea et al., 2004) data sets. The Senseval-2 data set⁴ is composed of 3 documents that contain 2473 ambiguous words (1136 nouns, 581 verbs, 457 adjectives, 299 adverbs), while the Senseval-3 data set⁴ is composed of 3 documents that contain 2081 ambiguous words (951 nouns, 751 verbs, 364 adjectives, 15 adverbs).

4.2 Parameter Tuning

As Schwab et al. (2013a), we tune our parameters on the first document of SemEval 2007. We first set the value of the internal parameter c to 20 without specifically tuning it. Using this value for c gives us a reasonable amount of configuration choices for the subsequent steps, without using too

³<http://nlp.cs.swarthmore.edu/semeval/tasks/index.php>

⁴<http://web.eecs.umich.edu/~mihalcea/downloads.html>

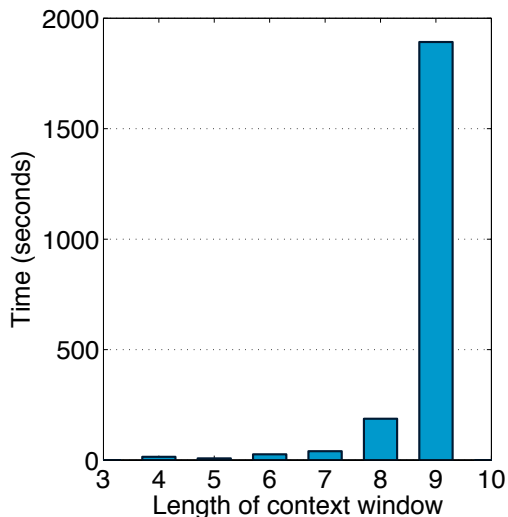


Figure 1: The running times (in seconds) of ShotgunWSD based on sense embeddings, on the first document of SemEval 2007, using various context window lengths $n \in \{4, 5, 6, 7, 8, 9\}$. The reported times were measured on a computer with Intel Core i7 3.4 GHz processor and 16 GB of RAM using a single Core.

much space and time. For tuning the parameters n and k , we employ sense embeddings for computing the semantic relatedness score. We begin by tuning the length of the context windows n . It is important to note that the upper bound accuracy of ShotgunWSD is given by the brute-force algorithm that explores every possible configuration of senses. Intuitively, we will get closer and closer to this upper bound as we use longer and longer context windows. However, the main decision factor is the time, which grows exponentially with respect to the length of the windows. Figure 1 illustrates the time required by our algorithm to disambiguate the first document in SemEval 2007, for increasing window lengths in the range 4-9. The algorithm runs in about 15 seconds for $n = 4$ and in about 1892 seconds for $n = 9$, so it becomes nearly 120 times slower from using context windows of length 4 to context windows of length 9. As the algorithm runs in a reasonable amount of time for $n = 8$ (187 seconds), we choose to use context windows of 8 words throughout the rest of the experiments.

The parameter k has almost no influence on the running time of the algorithm, so we tune this parameter with respect to the F_1 score obtained on the first document of SemEval 2007. We try out several values of k in the set $\{1, 3, 5, 10, 15, 20\}$

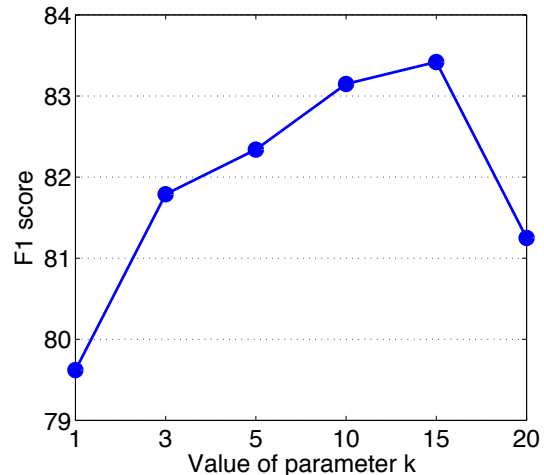


Figure 2: The F_1 scores of ShotgunWSD based on sense embeddings on the first document of SemEval 2007, using different values for the parameter $k \in \{1, 3, 5, 10, 15, 20\}$.

and the results are shown in Figure 2. The best F_1 score (83.42%) is obtained for $k = 15$. Hence, we choose to assign the final sense for each word using a majority vote based on the top 15 configurations.

To summarize, all the results of ShotgunWSD on SemEval 2007, Senseval-2 and Senseval-3 are reported using $n = 8$ and $k = 15$. We hereby note that better results in terms of accuracy can probably be obtained by trying out other values for these parameters on each data set. However, tuning the parameters on a single document from SemEval 2007 ensures that we avoid overfitting to a particular data set.

4.3 Results on SemEval 2007

We first conduct a comparative study on the SemEval 2007 coarse-grained English all-words task in order to evaluate our ShotgunWSD algorithm. As described in Section 3.1, we use two different approaches for computing the semantic relatedness scores, namely extended Lesk and sense embeddings. We compare our two variants of ShotgunWSD with several algorithms presented in (Schwab et al., 2012; Schwab et al., 2013a), namely a Genetic Algorithm, Simulated Annealing, and Ant Colony Optimization. We also include in the comparison an approach based on sense embeddings (Chen et al., 2014). All the approaches comprised in the evaluation are unsupervised. We compare them with the MCS baseline which is based on human annotations. The F_1

Method	F_1 Score
Most Common Sense	78.89%
Genetic Algorithms (Schwab et al., 2013a)	74.53%
Simulated Annealing (Schwab et al., 2013a)	75.18%
Ant Colony (Schwab et al., 2013a)	79.03%
S2C Unsupervised (Chen et al., 2014)	75.80%
ShotgunWSD + Extended Lesk	79.15%
ShotgunWSD + Sense Embeddings	79.68%

Table 1: The F_1 scores of various unsupervised state-of-the-art WSD approaches, compared to the F_1 scores of ShotgunWSD based on the extended Lesk measure and ShotgunWSD based on sense embeddings, on the SemEval 2007 coarse-grained English all-words task. The results reported for both ShotgunWSD variants are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations.

scores on SemEval 2007 are presented in Table 1. Among the state-of-the-art methods, it seems that the Ant Colony Optimization algorithm, based on a weighted voting scheme (Schwab et al., 2013a), is the only method able to surpass the MCS baseline. The unsupervised S2C approach gives lower results than the MCS baseline, but Chen et al. (2014) report better results in a semi-supervised setting. Both variants of ShotgunWSD yield better results than the MCS baseline (78.89%) and the Ant Colony Optimization algorithm (79.03%). Indeed, we obtain an F_1 score of 79.15% when using the extended Lesk measure and an F_1 score of 79.68% when using sense embeddings. We can also point out that ShotgunWSD gives slightly better results when sense embeddings are used instead of the extended Lesk method.

An important remark is that we have tuned the parameter k on the first document included in the test set, following the same evaluation procedure as Schwab et al. (2013a). Although this brings us to a fair comparison with Schwab et al. (2013a), it might also raise suspicions of overfitting the parameter k to the test set. Hence, we have tested all values of k in $\{1, 3, 5, 10, 15, 20\}$ for ShotgunWSD based on word embeddings, and we have always obtained results above 79%, with the top score of 79.77% for $k = 10$.

4.4 Results on Senseval-2

We compare the two alternative forms of ShotgunWSD with the MCS baseline, the MCS estimation method of Bhingardive et al. (2015) and the extended Lesk measure (Torres and Gelbukh,

Method	F_1 Score
Most Common Sense	60.10%
MCS Estimation (Bhingardive et al., 2015)	52.34%
Extended Lesk (Torres and Gelbukh, 2009)	54.60%
ShotgunWSD + Extended Lesk	55.78%
ShotgunWSD + Sense Embeddings	57.55%

Table 2: The F_1 scores of an unsupervised WSD approach and the extended Lesk measure, compared to the F_1 scores of ShotgunWSD based on the extended Lesk measure and ShotgunWSD based on sense embeddings, on the Senseval-2 English all-words data set. The results reported for both ShotgunWSD approaches are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations.

2009) on the Senseval-2 English all-words data set. As shown in Table 2, the ShotgunWSD based on sense embeddings obtains an F_1 score that is almost 5% better than the F_1 score of Bhingardive et al. (2015), while the ShotgunWSD based on extended Lesk gives an F_1 score that is around 1% better than the F_1 score reported by Torres and Gelbukh (2009). It is important to note that Torres and Gelbukh (2009) apply the extended Lesk measure by performing the brute-force search at the sentence level, hence it is not surprising that we are able to obtain better results. However, our best ShotgunWSD approach (57.55%) is still under the MCS baseline (60.10%).

4.5 Results on Senseval-3

Method	F_1 Score
Most Common Sense	62.30%
MCS Estimation (Bhingardive et al., 2015)	43.28%
Extended Lesk (Torres and Gelbukh, 2009)	49.60%
ShotgunWSD + Extended Lesk	57.89%
ShotgunWSD + Sense Embeddings	59.82%

Table 3: The F_1 scores of an unsupervised WSD approach and the extended Lesk measure, compared to the F_1 scores of ShotgunWSD based on the extended Lesk measure and ShotgunWSD based on sense embeddings, on the Senseval-3 English all-words data set. The results reported for both ShotgunWSD approaches are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations.

We also compare the two variants of ShotgunWSD with the MCS baseline, the MCS estimation method of Bhingardive et al. (2015) and the extended Lesk measure (Torres and Gelbukh, 2009)

on the Senseval-3 English all-words data set. The F_1 scores are presented in Table 3. The empirical results show that both ShotgunWSD variants give considerably better results compared to the MCS estimation method of Bhingardive et al. (2015). By using sense embeddings in a completely different way than Bhingardive et al. (2015), we are able to report an F_1 score of 59.82%, which is much closer to the MCS baseline (62.30%). With an F_1 score of 57.89%, the ShotgunWSD based on the extend Lesk measure brings an improvement of 8% over the extended Lesk algorithm applied at the sentence level (Torres and Gelbukh, 2009).

4.6 Discussion

Considering all the experiments, we can conclude that ShotgunWSD gives better results (around 1%) when sense embeddings are used instead of the extended Lesk method. On one of the data sets, ShotgunWSD yields better performance than the MCS baseline. It is important to underline that the strong MCS baseline cannot be used in practice, since human input is required to indicate which sense of a word is the most frequent in a given text (a word's dominant sense will vary across domains and text genres). Corpora used for the evaluation of WSD methods usually contain this kind of annotations, but the MCS baseline will not work outside the annotated data. Therefore, we consider important even slightly outperforming the MCS baseline. Overall, our algorithm compares favorably to other state-of-the-art unsupervised WSD methods (Schwab et al., 2013a; Chen et al., 2014; Bhingardive et al., 2015) and to the extended Lesk measure (Banerjee and Pedersen, 2002; Torres and Gelbukh, 2009).

Regarding the performance of our algorithm, an interesting question that arises is how much does the assembly phase help. We look to investigate this further in future work, but we can carry out a small experiment to provide a quick answer to this question. We consider the ShotgunWSD variant based on sense embeddings without changing its parameters, and we remove the assembly phase completely. Therefore, the algorithm will no longer produce configurations of length greater than 8, as the parameter n is set to 8. We have evaluated this stub algorithm on SemEval 2007 and we have obtained a lower F_1 score (77.61%). This indicates that the assembly phase in Algorithm 1 boosts the performance by nearly 2%. More ex-

periments are required to make sure that the performance boost is consistent across data sets.

5 Conclusions and Future Work

In this paper, we have introduced a novel unsupervised global WSD algorithm inspired by the Shotgun genome sequencing technique (Anderson, 1981). Compared to other bio-inspired WSD methods (Schwab et al., 2012; Schwab et al., 2013a), our algorithm has only two parameters. Furthermore, our algorithm is deterministic, obtaining the same result for a given set of parameters and input document. The empirical results indicate that our algorithm can obtain better performance than other state-of-the-art unsupervised WSD methods (Schwab et al., 2013a; Chen et al., 2014; Bhingardive et al., 2015). Although the fact that ShotgunWSD is deterministic brings several advantages, it is also a key difference from our source of inspiration, Shotgun sequencing, which is a non-deterministic technique.

In future work, we aim to investigate if training sense embeddings instead of deriving them from pre-trained word embeddings could yield better accuracy. Another promising direction is to compute the semantic relatedness of sense configurations based on the sum of sense tuples instead of sense pairs. An approach to combine the two semantic relatedness approaches independently used by ShotgunWSD, namely the extended Lesk measure and sense embeddings, is also worth exploring in the future.

Acknowledgments

We thank the reviewers for their helpful suggestions. We also thank Alexandru I. Tomescu from the University of Helsinki for insightful comments about the Shotgun sequencing technique.

References

- Eneko Agirre and Philip Glenny Edmonds. 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer.
- Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random Walks for Knowledge-based Word Sense Disambiguation. *Computational Linguistics*, 40(1):57–84, March.
- Stephen Anderson. 1981. Shotgun DNA sequencing using cloned DNase I-generated fragments. *Nucleic Acids Research*, 9(13):3015–3027.

- Satanjeev Banerjee and Ted Pedersen. 2002. An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. *Proceedings of CICLing*, pages 136–145.
- Satanjeev Banerjee and Ted Pedersen. 2003. Extended Gloss Overlaps As a Measure of Semantic Relatedness. *Proceedings of IJCAI*, pages 805–810.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, March.
- Simon Bennett. 2004. Solexa Ltd. *Pharmacogenomics*, 5(4):433–438, June.
- Sudha Bhingardive, Dharendra Singh, Rudramurthy V, Hanumant Harichandra Redkar, and Pushpak Bhat-tacharyya. 2015. Unsupervised Most Frequent Sense Detection using Word Embeddings. *Proceedings of NAACL*, pages 1238–1243.
- Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. *Proceedings of EMNLP*, pages 61–72.
- Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A Unified Model for Word Sense Representation and Disambiguation. *Proceedings of EMNLP*, pages 1025–1035, October.
- Adrian-Gabriel Chifu and Radu Tudor Ionescu. 2012. Word sense disambiguation to improve precision for ambiguous queries. *Central European Journal of Computer Science*, 2(4):398–411.
- Adrian-Gabriel Chifu, Florentina Hristea, Josiane Mothe, and Marius Popescu. 2014. Word sense discrimination in information retrieval: a spectral clustering-based approach. *Information Processing & Management*, 1(2):16–31, July.
- Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of ICML*, pages 160–167.
- Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. *Proceedings of SENSEVAL*, pages 1–5.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.
- Florentina Hristea, Marius Popescu, and Monica Dumitrescu. 2008. Performing word sense disambiguation at the border between unsupervised and knowledge-based techniques. *Artificial Intelligence Review*, 30(1-4):67–86.
- Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. *Proceedings of ACL*, pages 897–907, August.
- Sorin Istrail, Granger G. Sutton, Liliana Florea, Aaron L. Halpern, Clark M. Mobarry, Ross Lip-pert, Brian Walenz, Hagit Shatkay, Ian Dew, Jason R. Miller, Michael J. Flanigan, Nathan J. Edwards, Randall Bolanos, Daniel Fasulo, Bjarni V. Halldorsson, Sridhar Hannenhalli, Russell Turner, Shibu Yooseph, Fu Lu, Deborah R. Nusskern, Bixiong Chris Shue, Xiangqun Holly Zheng, Fei Zhong, Arthur L. Delcher, Daniel H. Huson, Saul A. Kravitz, Laurent Mouchard, Knut Reinert, Karin A. Remington, Andrew G. Clark, Michael S. Waterman, Evan E. Eichler, Mark D. Adams, Michael W. Hunkapiller, Eugene W. Myers, and J. Craig Ven-ter. 2004. Whole Genome Shotgun Assembly and Comparison of Human Genome Assemblies. *Proceedings of the National Academy of Sciences*, 101(7):1916–1921.
- Mathieu Lafourcade and Frédéric Guinand. 2010. Artificial Ants for Natural Language Processing. In N. Monmarch, F. Guinand, and P. Siarry, editors, *Artificial Ants. From Collective Intelligence to Real-life Optimization and Beyond*, chapter 20, pages 455–492. Wiley.
- Michael Lesk. 1986. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. *Proceedings of SIGDOC*, pages 24–26.
- V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reverseals. *Cybernetics and Control Theory*, 10(8):707–710.
- Rada Mihalcea, Timothy Chklovski, and Adam Kilgar-riff. 2004. The Senseval-3 English Lexical Sample Task. *Proceedings of SENSEVAL-3*, pages 25–28, July.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositional-ity. *Proceedings of NIPS*, pages 3111–3119.
- George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, November.
- Roberto Navigli, Kenneth C. Litkowski, and Orin Har-graves. 2007. SemEval-2007 Task 07: Coarse-grained English All-words Task. *Proceedings of SemEval*, pages 30–35.
- Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10:1–10:69, February.
- Kiem-Hieu Nguyen and Cheol-Young Ock. 2013. Word sense disambiguation as a traveling salesman problem. *Artificial Intelligence Review*, 40(4):405–427.
- Ravi K. Patel and Mukesh Jain. 2012. NGS QC Toolkit: A Toolkit for Quality Control of Next Generation Sequencing Data. *PLoS ONE*, 7(2):1–7, 02.

- Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using Measures of Semantic Relatedness for Word Sense Disambiguation. *Proceedings of CICLing*, pages 241–257.
- Laura Plaza, Antonio Jose Jimeno-Yepes, Alberto Diaz, and Alan R. Aronson. 2011. Studying the correlation between different word sense disambiguation methods and summarization effectiveness in biomedical texts. *BMC Bioinformatics*, 12:355–367.
- Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.
- Didier Schwab, Jérôme Goulian, Andon Tchechmedjiev, and Hervé Blanchon. 2012. Ant Colony Algorithm for the Unsupervised Word Sense Disambiguation of Texts: Comparison and Evaluation. *Proceedings of COLING*, pages 2389–2404, December.
- Didier Schwab, Jérôme Goulian, and Andon Tchechmedjiev. 2013a. Worst-case Complexity and Empirical Evaluation of Artificial Intelligence Methods for Unsupervised Word Sense Disambiguation. *International Journal of Engineering and Technology*, 8(2):124–153, August.
- Didier Schwab, Andon Tchechmedjiev, and Jérôme Goulian. 2013b. GETALP: Propagation of a Lesk Measure through an Ant Colony Algorithm. *Proceedings of SemEval*, 1(2):232–240, June.
- Chiraag Sumanth and Diana Inkpen. 2015. How much does word sense disambiguation help in sentiment analysis of micropost data? *Proceedings of WASSA*, pages 115–121, September.
- Sulema Torres and Alexander Gelbukh. 2009. Comparing Similarity Measures for Original WSD Lesk Algorithm. *Research in Computing Science*, 43:155–166.
- R. V. Vidhu Bhala and S. Abirami. 2014. Trends in word sense disambiguation. *Artificial Intelligence Review*, 42(2):159–171.
- Karl V. Voelkerding, Shale A. Dames, and Jacob D. Durtschi. 2009. Next Generation Sequencing: From Basic Research to Diagnostics. *Clinical Chemistry*, 55(4):41–47.

LanideNN: Multilingual Language Identification on Character Window

Tom Kocmi and Ondřej Bojar

Charles University, Faculty of Mathematics and Physics

Institute of Formal and Applied Linguistics

{kocmi,bojar}@ufal.mff.cuni.cz

Abstract

In language identification, a common first step in natural language processing, we want to automatically determine the language of some input text. Monolingual language identification assumes that the given document is written in one language. In multilingual language identification, the document is usually in two or three languages and we just want their names. We aim one step further and propose a method for textual language identification where languages can change arbitrarily and the goal is to identify the spans of each of the languages.

Our method is based on Bidirectional Recurrent Neural Networks and it performs well in monolingual and multilingual language identification tasks on six datasets covering 131 languages. The method keeps the accuracy also for short documents and across domains, so it is ideal for off-the-shelf use without preparation of training data.

1 Introduction

The World Wide Web is an ever growing source of textual data, especially data generated by web users. As more people get access to the web, more languages and dialects start to appear and need to be processed. In order to be able to use such data for further natural language processing (NLP) tasks, we need to know in which languages they were written. Language identification is thus a key component for both building various NLP resources from the web and also for running many web services.

Techniques of language identification can rely on handcrafted rules, usually of high precision but

low coverage, or data-driven methods that learn to identify languages based on sample texts of sufficient quantity.

In this paper, we present a data-driven method for language identification based on bidirectional recurrent neural networks called *LanideNN* (language identification by neural networks, NN). The model is trained on character sliding window of input texts with the goal of assigning a language to each character. We show that the method is applicable for a large number of languages and across text domains without any adaptation and that it performs well in monolingual (one language per document) as well as multilingual (a few languages per document) language identification tasks. Also, the performance does not drop with shorter texts.

The paper is structured as follows. In Section 2, we briefly review current approaches to language identification. Section 3 introduces our method, including the technical details of the neural network architecture. For the training of our model, we collect and manually clean a new dataset, as described in Section 4. The model is evaluated on standard test sets for monolingual (Section 5) as well as multilingual (Section 6) language identification. Section 7 illustrates the behavior of our method in the motivating setting: identifying languages in short texts. We conclude and summarize our plans in Section 8.

2 Related Work

Of the many possible approaches to language identification Hughes et al. (2006), character n -gram statistics are among the most popular ones. Cavnar et al. (1994) were probably the first; they used the 300 most frequent character n -grams (with n ranging from 1 to 5, as is also typically used in other works). All the n -gram-based ap-

proaches differ primarily in the calculation of the distance between the n -gram profile of the training and test text (Selamat, 2011; Yang and Liang, 2010), or by using additional features on top of the n -gram profiles (Padma et al., 2009; Carter et al., 2013). One of the fairly robust definitions of the distance (or similarity) was proposed by Choong et al. (2009) who simply check the proportion of n -gram types seen in the tested document of the most frequent n -gram types extracted from training documents for each language. The highest-scoring language is then returned.

Hughes et al. (2006) mention a number of freely available tools at that time. Since then, one aspect of the tools became also important: the number of languages covered.

The language identification tool CLD2¹ by Google detects 80 languages and uses a Naive Bayes classifier, treating specifically unambiguous scripts such as Greek and using either character unigrams (Han and similar scripts) or fourgrams.

Another popular tool is Langid.py by Lui and Baldwin (2012), covering 97 languages out of the box. Langid.py relies on Naive Bayes classifier with a multinomial event model and mixture of byte n -grams for training. The tool includes tokenization and fast feature extraction using Aho-Corasick string matching.

To our knowledge, and also according to the survey by Garg et al. (2014), neural networks have not been used often for language identification so far. One exception is Al-Dubaei et al. (2010), who combine a feed-forward network classifier with wavelet transforms of feature vectors to identify English and Arabic from the Unicode representation of words, sentences or whole documents. The benefit of NN in this setting is not very clear to us because English and Arabic can be distinguished by the script. During writing of this paper, we have found a new pre-print paper (Jaech et al., 2016) which handles language identification with NN. Specifically, they employ Convolutional Neural Networks followed by Recurrent Neural Networks. Their approach labels text on the word level, which is problematic in languages without clear word delimiters. In comparison with our model, they need to pre-process the data and break long words into smaller chunks, whereas we simply use text without any preprocessing.

In practice, several tools are often used at once,

¹<https://github.com/CLD2Owners/cld2>

with some form of majority voting. For example, Twitter internal language detector uses their in-house tool along with CLD2 and Langid.py, and this triple agreement is reported to make less than 1% of errors.²

Multilingual language identification, i.e. identification of the set of languages used in a document, is a less common task, explored e.g. by Lui et al. (2014) who use a generative mixture model on multilingual documents and establish the relative proportion of languages used. Character n -grams again serve as features, selected by information gain.

Solorio et al. (2014) organized a shared task in language identification at the word level. This matches our aim, but the task included only four language pairs and more importantly, the dataset was collected from Twitter and for copyright reasons it is not available any more.

3 Proposed Method

The method we propose is designed for short text without relying on document boundaries. Obviously, if documents are known and if they can be assumed to be monolingual, this additional knowledge should not be neglected. For the long term, we however aim at a streamlined processing of noisy data genuinely appearing in multilingual environments. For instance, our method could support the study of switching of languages (“code switching”) in e-mails or other forms of conversation, or to analyse various online media such as Twitter, see e.g. Montes-Alcalá (2007) or Solorio et al. (2014).

Our model takes source letters as input and provides a language label for each of them. Whenever we need to recognize the language of a document, we take the language assigned by our model to the majority of letters.

The goal of attributing a language tag to the smallest text units is one of the reasons why we decided to use neural networks and designed the model to provide a prediction at every time step without much overhead.

In the rest of this section, we explain the architecture and training methods of the model.

²<https://blog.twitter.com/2015/evaluating-language-identification-performance>

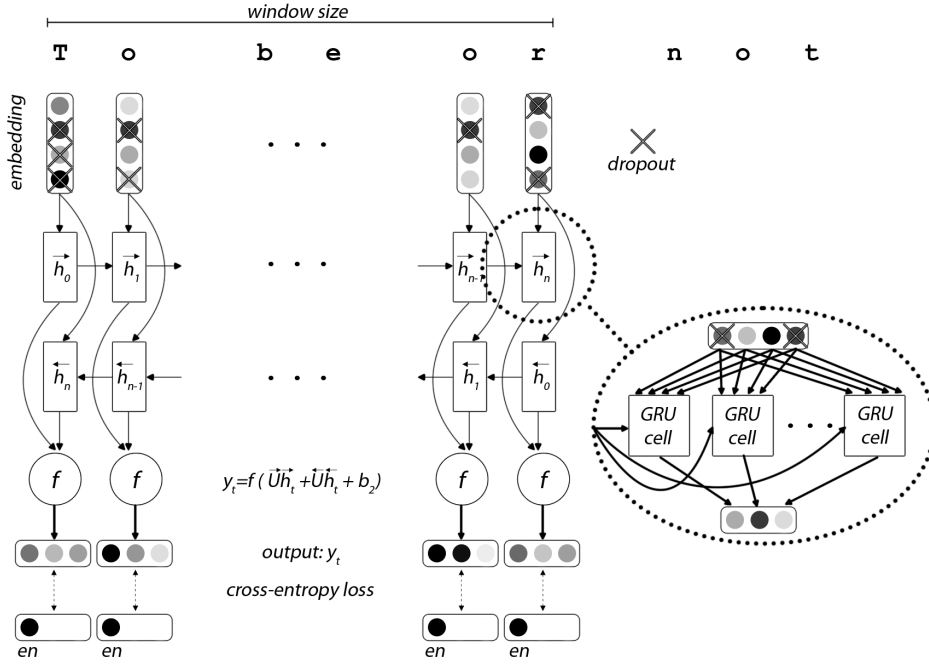


Figure 1: Illustration of our model LanideNN.

3.1 Bidirectional Recurrent Neural Networks

A recurrent neural network *RNN* (Elman, 1990) is a variant of neural networks with recurrent connections in time. In principle, the history information available to an RNN is not limited (subject to a processing window, if used), so the network can condition its output on features from a long distance. The LSTM, one of the variants of RNN, makes it particularly suitable for sequential prediction tasks with arbitrary time dependencies, as shown by Hochreiter and Schmidhuber (1997).

In this work, we use the Elman-type network, where the hidden layer h_t at a time step t is computed based on the current input layer x_t and the previous state of the hidden layer h_{t-1} . The output y_t is then derived from the h_t by applying the softmax function f . More formally:

$$h_t = \tanh(Wx_t + Vh_{t-1} + b_1) \quad (1)$$

$$y_t = f(Uh_t + b_2) \quad (2)$$

where U , V and W are connection weights to be computed in training time and bias vectors b_1 and b_2 .

With the above definition, the RNN has access only to information preceding the current position in the text. In our setting, the rest of the text (in a fixed-size window) is available, so we want to allow the model to use also future information, i.e.

letters following the currently examined one. We therefore define a second RNN which reads the input from the end to the beginning, changing the definition to:

$$\vec{h}_t = \tanh(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b}_1) \quad (3)$$

$$\overleftarrow{h}_t = \tanh(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b}_1) \quad (4)$$

$$y_t = f(\overrightarrow{U}\vec{h}_t + \overleftarrow{U}\overleftarrow{h}_t + b_2) \quad (5)$$

where the left and right arrows indicate the direction of network.

The simple unit with only tanh non-linearity is difficult to train and therefore we have selected the Gated Recurrent Unit (GRU), recently proposed by Cho et al. (2014), as a replacement. We also considered Long Short-Term Memory cells (LSTM) but they achieved slightly worse results in our setting. This changes equations (1), (3) and (4). The proper equations for the GRU can be found in Cho et al. (2014).

The model outputs a probability distribution over all language tags. In order to determine the language of a character, we take the tag with the maximum value.

The complete model is sketched in Figure 1.

3.2 Training, Embeddings and Dropout

We train the model using the first-order stochastic gradient descent method Adam (Kingma and Ba, 2015). Our training criterion is the cross-entropy loss function³.

We represent each Unicode character using an e -dimensional real valued vector, analogously to word embeddings of Collobert et al. (2011). The character embeddings are initialized randomly and are trained together with the rest of the network.

To prevent overfitting, we use dropout (Srivastava et al., 2014) during model training on the character embedding layer⁴. The key idea is to randomly drop (avoid updating of) connections. This prevents neurons from co-adapting too much, i.e. starting to depend on outputs of other neurons too much, which is a typical symptom of overfitting to training data.

3.3 Model Design

Our model operates on a window of 200 characters of input text, i.e. individual letters, encoded in Unicode. Each character corresponds to one time step of the BiRNN in the respective direction, see Figure 1. The model classifies each character separately, but quickly learns to classify neighbouring characters with the same label.

For documents longer than the window size, we simply move to the next window without any overlap. The last window (or the only window if the document were too short) is filled with a padding character, so the network always works on windows of the same size.

We set e , the size of the embedding layer, to 200. The BiRNN uses a single hidden layer of 500 GRU cells for each direction.

The main model was trained for over 530,000 steps (each step is the processing of one batch of inputs) on a single core of the GeForce GTX Titan Z GPU. The training took around 5 days. The stopping criterion for the training was the error on a development set.

4 Training Data

Our goal is to develop an off-the-shelf language recognizer, with no need for retraining by the user and covering as many languages as possible. Finding suitable training data is thus an important part

³We set the learning rate to 0.0001 and train with the batch size of 64 windows.

⁴We set the dropout to the probability of 0.5 as customary.

afr, amh, ara, arg, asm, ast, aze, bak, bcl, bel, ben, ber, bpy, bre, bul, cat, ceb, ces, che, chv, cos, cym, dan, deu, div, ekk, ell, eng, est, eus, fas, fin, fra, fry, gla, gle, glg, gom, gsw, guj, hat, heb, hif, hin, hrv, hsb, hun, hye, ido, ilo, ina, ind, isl, ita, jav, jpn, kal, kan, kas, kat, kaz, kir, kor, kur, lat, lav, lim, lit, ltz, lug, lus, mal, mar, min, mkd, mlg, mlt, mon, mri, msa, nds, nep, new, nld, nno, nor, nso, oci, ori, oss, pam, pan, pms, pnb, pol, por, pus, roh, ron, rus, sah, scn, sin, slk, slv, sna, som, spa, sqi, srp, sun, swa, swe, tam, tat, tel, tgk, tgl, tha, tur, uig, ukr, urd, uzb, vec, vie, vol, wln, yid, zho, zul

Figure 2: The 131 languages (and HTML) recognized by our system.

of the endeavour.

We start from Wikipedia, as crawled and converted to a large multilingual corpus W2C by Majliš and Žabokrtský (2012). W2C contains 106 languages but we had to exclude a few of them⁵ because they contained too little non-repeating text.

We then focussed on finding corpora with at least some languages not covered by the already collected data. Those corpora were added whole, including languages that we already had, to improve the variety of our collection. We made use of the following ones:

Tatoeba⁶ is a collection of simple sentences for language learners. Tatoeba contains sentences in 307 languages, but for most languages it has only a few hundred sentences.

Leipzig corpora collection (Quasthoff et al., 2006) covers 220 languages with newspaper text and other various texts collected from the web.

EMILLE (Baker et al., 2002) contains texts in 14 Indian languages and English.

Haitian Creole training data (Callison-Burch et al., 2011) were prepared by the organizers of WMT11 shared task on machine translation of SMS messages sent to an emergency hotline in the aftermath of the 2010 Haitian earthquake. We used only the official documents from the training data, not the actual SMS messages because they contained a lot of noise.

⁵Specifically, HAT, IDO, MGL, MRI, VOL, as identified by ISO language codes.

⁶<http://tatoeba.org/>

Test Set	Documents	Languages	Encoding	Document Length (bytes)	Avg. # characters
EuroGov	1500	10	1	17460.5 ± 39353.4	17037.3
TCL	3174	60	12	2623.2 ± 3751.9	1686.1
Wikipedia	4963	67	1	1480.8 ± 4063.9	1314.2

Table 1: Summary of testsets for monolingual language identification.

Additionally, we wanted our tool to distinguish HTML tags in the data, since they are the most frequent markup that needs to be separated from the processed data. Therefore, we have downloaded several Github projects in HTML and collected all strings enclosed with angle brackets, as a rather permissive approximation of HTML tags. We have dropped tags which were too long and we put each tag on a separate line. We have not deduplicated them for the training set.

The cleanup of the collected data was mostly manual. We deduplicated each of the sources by dropping identical lines, regardless of what lines correspond to in the individual sources (words, phrases, sentences or even paragraphs). We inspected data files for individual languages and removed lines containing English for languages not using Latin script. We also removed Cyrillic characters from a few languages that should not contain them. This was done mostly in W2C corpora.

For the final dataset, we mixed all sources for a given language at the line level, keeping only languages with more than 500k characters in total. Since the resources for some languages were huge, we decided to set an upper bound on the number of characters per language. In order to roughly reflect the distribution of languages in the world, we divided languages into three groups based on the number speakers of the language according to Wikipedia. The first group were languages with more than 75M speakers, the second with more than 10M speakers and the third group contained the rest. For the first group, we allowed at most 10M characters in the training set, the second group was capped at 5M characters and the third group was allowed only 1M characters per language at most.⁷

In total, our final training set includes 131 + 1 (HTML) languages, see Figure 2.

We divide the corpus into non-overlapping training, development and test sections. We re-

⁷Higher-quality sources such as Tatoeba are generally smaller and since we mixed the sources by interleaving their lines, these smaller sources were likely included in full.

leased the test set ⁸ but the training part cannot be publicly released because of the restrictive permissions of some of the sources used. The test section is limited to short text. It contains 100 lines for each of the 131 languages (HTML is not included), with the average line length of 142.3 characters.

Each line of the dataset starts with an ISO-3 label of the language presented on that line. All lines were shuffled.

For training and testing, the language labels as well as all line breaks must be ignored, otherwise the model could learn to set language boundaries at the new line character. After dropping all line breaks, we obtain a multilingual text.

This way, we simulate a multilingual text and our algorithm has to learn to identify language boundaries without relying on any particular symbol. We are aware of the fact that the original segmentation of the corpora affects where these language switches are expected, and this will mostly correspond to sentence boundaries.

5 Monolingual Language Identification

Most of related research is focused on monolingual language identification, i.e. recognizing the single language of an input document.

We compare our method in this setting with several other algorithms on the dataset presented by Baldwin and Lui (2010). The dataset consists of 3 different test sets, each containing a different number of languages, styles and document lengths collected from different sources, see Table 1 for details:

EuroGov contains texts in Western European languages from European government resources.

TCL was extracted by the Thai Computational Linguistics Laboratory in 2005 from online news sources and the test set also contains multiple file encodings. Since our method assumes Unicode input, we converted TCL to Unicode encoding.

⁸<https://ufal.mff.cuni.cz/tom-kocmi/lanidenn>

System	Trained on	Supported languages	EuroGov	TCL	Wikipedia
LangDetect*	Wikipedia	53	.992 ⁹	.818	.867
TextCat*	TextCat Dataset	75	.941	.605	.706
CLD*	unknown	64	.983	.732	.831
Langid.py*	Lui and Baldwin (2011)	97	.987	.904	.913
Langid.py	Lui and Baldwin (2011)	97	.987	.931	.913
CLD2	unknown	83	.979	.837	.854
Our model	Our dataset	136	.977	.954	.893

Table 2: Results of monolingual language identification on the Baldwin and Lui (2010) test set. Entries marked with “*” are accuracies reported by Lui and Baldwin (2012), the rest are our measurements.

Wikipedia are texts collected from a Wikipedia dump.

Table 2 summarizes the accuracies of several algorithms on the three test sets. For some algorithms, we report values as presented by Lui and Baldwin (2012) without re-running⁹. We re-ran the Langid.py as the best algorithm out of them, and got the same results except for the TCL test-set, where we got better results than reported by Lui and Baldwin (2012). After a discussion with the authors, we believe the re-run benefited from the conversion of all texts to Unicode.

We compare our method with two top language recognizers, Langid.py and CLD2. Our model is trained on more languages and we do not restrict it to the languages included in the test set, so we may be losing on detailed dialect labels.

Despite the considerably higher number of languages covered, our model performs reasonably close to the competitors on EuroGov and Wikipedia and best on TCL.

5.1 Short-Text Language Identification

In order to demonstrate the ability of our method to identify language of very short texts such as tweets, search queries or user messages, we wanted to use an existing corpus, such as the one released by Twitter.¹⁰ Unfortunately, the corpus contains only references to the actual tweets and most of them are no longer available. We thus have to rely on our own test set, as described in Section 4.

Results on short texts are reported in Table 3. The two other systems, Langid.py and CLD2 cover fewer languages and they were trained on texts unrelated to our collection of data. It is there-

⁹We should mention that LangDetect used EuroGov as a validation set, so its score on this test set is not reliable.

¹⁰<https://blog.twitter.com/2015/evaluating-language-identification-performance>

System	All languages	Common languages
Langid.py	.567	.912
CLD2	.545	.891
Our model	.950	.955

Table 3: Results on our test set for short texts. The first column shows accuracy over all 131 languages and the second column shows accuracy over languages that all systems have in common.

ind↔msa	64	ekk↔est	36	bak↔tat	28
hrv↔srp	17	glg↔por	17	nno↔nor	16
ast↔spa	15	fas↔pus	13	ces↔slk	13
hrv↔slv	10	dan↔nor	10	nep↔new	8
aze↔tur	7	mar↔new	6	ceb↔tgl	6
cat↔spa	6	arg↔spa	6	fra↔oci	5

Table 6: Most frequent confusions on our test set.

fore not surprising that they perform much worse when averaged over all languages.

For a fairer comparison, we report also accuracies on a restricted version of the test set that included only languages supported by all the three tested tools. Both our competitors are meant to be generally applicable, so they should (and do) perform quite well. Our system nevertheless outperforms them, reaching the accuracy of 95.5. Arguably, we can be benefitting from having trained on (different) texts from the same sources as this test set.

Table 6 lists the most frequent misclassifications of our model on our test set (unordered language pairs) of the 13100 items in the test set. The most common error is confusing Indonesian with Modern Standard Arabic, which indicates some noise in our training data rather than difficulty of separating these two languages. The following pairs are expected: Standard Estonian (ekk) vs. Estonian (est, a macro language which includes Standard Estonian), Bashkir vs. Tatar, Croatian vs. Serbian, Asturian vs. Spanish, ...

Finally, our model is trained to distinguish also

System	Training set	P_M	R_M	F_M	P_μ	R_μ	F_μ
VRL (2010) *	ALTW2010	.497	.467	.464	.833	.826	.829
ALTW2010 winner *	ALTW2010	.718	.703	.699	.932	.931	.932
SEGLANG *	ALTW2010 - mono	.801	.810	.784	.866	.946	.905
LINGUINI *	ALTW2010 - mono	.616	.535	.513	.713	.688	.700
Lui et al. (2014) *	ALTW2010 - mono	.753	.771	.748	.945	.922	.933
Lui et al. (2014) our retrain	ALTW2010 - mono	.768	.716	.724	.968	.896	.931
Our model	ALTW2010 - mono	.819	.764	.779	.966	.964	.965
Our model	Our dataset	.709	.714	.695	.941	.941	.941

Table 4: Results of multilingual language identification on the ALTW2010 test set. * As reported by Lui et al. (2014)

System	P_M	R_M	F_M	P_μ	R_μ	F_μ
SEGLANG *	.809	.975	.875	.771	.975	.861
LINGUINI *	.853	.772	.802	.838	.774	.805
Lui et al. (2014) *	.962	.954	.957	.963	.955	.959
Lui et al. (2014) our retrain	.962	.963	.961	.963	.964	.963
Our model trained on WikipediaMulti	.962	.974	.966	.954	.974	.964
Our model trained on our dataset	.774	.778	.774	.949	.972	.961
Our model trained on our dataset, restricted	.966	.973	.966	.956	.973	.964

Table 5: Results of multilingual language identification on the WikipediaMulti test set. * As reported by Lui et al. (2014)

HTML as one additional language. We did not include HTML in our test corpus but to satisfy the requests of one of our reviewers, we checked the performance on our development corpus: only one Portuguese and one Yakut segment was classified as HTML and none of the 100 HTML segments were misclassified.

6 Multilingual Language Identification

In multilingual language identification, systems are expected to report the set of languages used in each input document. The evaluation criterion is thus macro- (M) or micro- (μ) averaged precision (P), recall (R) or F-measure (F).¹¹

We evaluate our model on two existing test sets for multilingual identification, ALTW2010 shared task and WikipediaMulti. We are mainly interested in the performance of our general model, trained on all our training data, on these test sets. But since both test sets come with training data, we also retrain our model to test its in-domain performance. We limit the training of these specific models to 140,000 training steps for ALTW2010 and 75,000 steps for WikiMulti, keeping other settings identical to the main model. Each training step amounts to the processing of 64 batches of 200 letters of input. The number of steps for both

¹¹Note that for comparability with results reported in other works, macro-averaged F-score is calculated as average over individual F-scores instead of the harmonic mean of P_M and R_M . F_M can thus fall out of the range between P_M and R_M .

tasks was established by testing the error on the development parts of the datasets.

To interpret the character-level predictions by our model for multilingual identification, we used the ALTW2010 development data to empirically set the threshold: if a language is predicted for more than 3% of characters in the document, it is considered as one of the languages of the document.

6.1 ALTW 2010 Shared Task

ALTW 2010 shared task (VRL, 2010) provided 10000 bilingual documents divided as follows: 8000 training, 1000 development and 1000 test documents.

The results on the 1000 test documents are in Table 4. For algorithms SEGLANG and LINGUINI, we only reproduce the results reported by Lui et al. (2014). We use the system by Lui et al. (2014) as a proxy for the comparison: we retrain their system and obtain results similar to those reported by the original authors. The differences are probably due to the Gibbs sampling used in their approach.

Some of the reported methods rely on the fact that the documents in the dataset are bilingual. Other methods, including ours, simply break the bilingual documents into the individual languages and train on this simplified training set. We indicate this by stating “ALTW2010 - mono” in Table 4.



Figure 3: Illustration of text partitioning. The black triangles indicate true boundaries of languages. The black part shows probability with which the language written in gray is detected and the gray part shows complement for the second language, since in this setup we restricted our model to use only the two languages in question. The misclassification of Italian and German as English in the last two examples may reflect increased noise in our English training data.

The main criterion of the ALTW2010 shared task was to maximize the micro-averaged F-score (F_μ). We see that our model trained on the ALTW2010 data outperforms all other models in this criterion (F_μ of .965) and so does our non-adapted version, reaching F_μ of .941.

6.2 WikipediaMulti

WikipediaMulti (Lui et al., 2014) is a dataset of artificially prepared multilingual documents, mixed from monolingual Wikipedia articles from 44 languages. Each of the artificial documents contains texts in $1 \leq k \leq 5$ randomly selected languages. The average document length is 5500 bytes. The training set consists of 5000 monolingual documents, the development set consists of 5000 multilingual documents and test set consists of 1000 documents for each value of k .

Table 5 shows that our model performs well, both when trained on the provided data and when trained on our training corpus. The model trained on our dataset performs slightly worse in F_μ , but if we simply prevent it from predicting languages

not present in the test set, the score gets on par with the adapted version, see the line labelled “restricted” in Table 5.

7 Text Partitioning

Figure 3 illustrates the behaviour of our model on text with mixed languages. We have selected very short (50–130 characters) and challenging segments where the languages mostly share the same script. Finding the boundary between languages written in different scripts is quite easy, as illustrated by the first example.

Only too late, we discovered that King and Abney (2013) provide a test set for word-level identification for 30 languages. We thus have to leave the evaluation of our model on this dataset for future.

8 Conclusion

We have developed a language identification algorithm based on bidirectional recurrent neural networks. The approach is designed for identifying

languages on a short texts, allowing to detect code switching including switches to formal markup languages like HTML.

We collected a dataset and trained our model to recognize considerably more languages than other state-of-the-art tools. Our algorithm and the trained model is provided for academic and personal use.¹²

Since there is no established dataset for the novel setting of text partitioning by language, we evaluated our model in several common tasks (monolingual and multilingual language identification for long and short texts) which were previously handled by separate algorithms. Our approach performs well, improving over the state of the art in several cases.

A number of things are planned: (1) improving the implementation, especially the speed of application of a trained model, (2) further extending the set of covered languages and possibly including more artificial or programming languages (e.g. JavaScript, PHP) or common formal notations (URLs, hashtags), (3) evaluating our method on the dataset by King and Abney (2013), possibly extending this dataset to include more languages, (4) training and testing the model on noisy texts like Tweets or forum posts, and (5) experimenting with other network architectures and approaches, possibly also training the model on bytes instead of Unicode characters.

9 Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 645452 (QT21), the grant GAUK 8502/2016, and SVV project number 260 333.

This work has been using language resources developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071)

References

Shawki A Al-Dubae, Nesar Ahmad, Jan Martinovic, and Vaclav Snasel. 2010. Language identification using wavelet transform and artificial neural network. In *Computational Aspects of Social Networks (CASoN), 2010 International Conference on*, pages 515–520. IEEE.

¹²<https://github.com/tomkocmi/LanideNN>

Paul Baker, Andrew Hardie, Tony McEnery, Hamish Cunningham, and Robert J Gaizauskas. 2002. Emille, a 67-million word corpus of indic languages: Data collection, mark-up and harmonisation. In *LREC*.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 229–237, Los Angeles, California, June. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July. Association for Computational Linguistics.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, 47(1):195–215.

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Chew Choong, Yoshiki Mikami, Chandrajith Ashuboda Marasinghe, and ST Nandasara. 2009. Optimizing n-gram order of an n-gram based language identification algorithm for 68 written languages. *International Journal on Advances in ICT for Emerging Regions (ICTer)*, 2(2).

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

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

Archana Garg, Vishal Gupta, and Manish Jindal. 2014. A survey of language identification techniques and applications. *Journal of Emerging Technologies in Web Intelligence*, 6(4):388–400.

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

- Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew Mackinlay. 2006. Reconsidering language identification for written language resources. In *Proceedings of LREC2006*, pages 485–488.
- Aaron Jaech, George Mulcaire, Shobhit Hathi, Mari Ostendorf, and Noah A. Smith. 2016. Hierarchical character-word models for language identification. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 84–93, Austin, TX, USA, November. Association for Computational Linguistics.
- Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1110–1119, Atlanta, Georgia, June. Association for Computational Linguistics.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*.
- Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 553–561, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.
- Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea, July. Association for Computational Linguistics.
- Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, 2:27–40.
- Martin Majliš and Zdeněk Žabokrtský. 2012. Language richness of the web. In *LREC*, pp. 2927–2934.
- Cecilia Montes-Alcalá. 2007. Blogging in Two Languages: Code-Switching in Bilingual Blogs. In Jonathan Holmquist, Augusto Lorenzino, and Lotfi Sayahi, editors, *Selected Proceedings of the Third Workshop on Spanish Sociolinguistics*, pages 162–170. Cascadilla Proceedings Project, Somerville, MA, USA.
- M. C. Padma, P. A. Vijaya, and P. Nagabhushan. 2009. Language identification from an indian multilingual document using profile features. In *2009 International Conference on Computer and Automation Engineering*, pages 332–335, March.
- Uwe Quasthoff, Matthias Richter, and Christian Biemann. 2006. Biemann c., corpus portal for search in monolingual corpora. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.
- Ali Selamat, 2011. *Improved N-grams Approach for Web Page Language Identification*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, Doha, Qatar, October. Association for Computational Linguistics.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January.
- NICTA VRL. 2010. Multilingual language identification: Altw 2010 shared task dataset. In *Australasian Language Technology Association Workshop 2010*, page 4.
- X. Yang and W. Liang. 2010. An n-gram-and-wikipedia joint approach to natural language identification. In *2010 4th International Universal Communication Symposium*, pages 332–339, Oct.

Cross-Lingual Word Embeddings for Low-Resource Language Modeling

Oliver Adams,^{♠♥} Adam Makarucha,[♠] Graham Neubig,[♣] Steven Bird,^{♥◇} Trevor Cohn[♥]

[♠]IBM Research – Australia

[♥]Computing and Information Systems, University of Melbourne

[♣]School of Computer Science, Carnegie Mellon University

[◇]International Computer Science Institute, University of California Berkeley

oliver.adams@gmail.com, adamjm@au1.ibm.com,

gneubig@cs.cmu.edu, {t.cohn, steven.bird}@unimelb.edu.au

Abstract

Most languages have no established writing system and minimal written records. However, textual data is essential for natural language processing, and particularly important for training language models to support speech recognition. Even in cases where text data is missing, there are some languages for which bilingual lexicons are available, since creating lexicons is a fundamental task of documentary linguistics. We investigate the use of such lexicons to improve language models when textual training data is limited to as few as a thousand sentences. The method involves learning cross-lingual word embeddings as a preliminary step in training monolingual language models. Results across a number of languages show that language models are improved by this pre-training. Application to Yongning Na, a threatened language, highlights challenges in deploying the approach in real low-resource environments.

1 Introduction

Most of the world’s languages are not actively written, even languages with an official writing system (Bird, 2011). This limits the available textual data to small quantities of phonemic transcriptions prepared by linguists. Since phonemic transcription is time-consuming, such data is scarce. This makes language modeling, which is a key tool for facilitating speech recognition of these languages, a difficult challenge. One of the touted advantages of neural network language models (NNLMs) is their ability to model sparse data (Bengio et al., 2003; Gandhe et al., 2014). However, despite the success of NNLMs

on large datasets (Mikolov et al., 2010; Sutskever et al., 2011; Graves, 2013), it remains unclear whether their advantages transfer to scenarios with extremely limited amounts of data.

Appropriate initialization of parameters in neural network frameworks has been shown to be beneficial across a wide variety of domains, including speech recognition, where unsupervised pre-training of deep belief networks was instrumental in attaining breakthrough performance (Hinton et al., 2012). Neural network approaches to a range of NLP problems have also been aided by initialization with word embeddings trained on large amounts of unannotated text (Frome et al., 2013; Zhang et al., 2014; Lau and Baldwin, 2016). However, in the case of extremely low-resource languages we do not have the luxury of this unannotated text.

As a remedy to this problem we focus on cross-lingual word embeddings (CLWEs), which learn word embeddings using information from multiple languages. Recent advances in CLWEs have shown that high quality embeddings can be learnt even in the absence of bilingual corpora by harnessing bilingual lexicons (Gouws and Søgaard, 2015; Duong et al., 2016). This is useful as some threatened and endangered languages have been subject to significant linguistic investigation, leading to the creation of high-quality lexicons, despite the dearth of transcriptions. For example, the training of a quality speech recognition system for Yongning Na, a Sino-Tibetan language spoken by approximately 40k people, is hindered by this lack of data (Do et al., 2014) despite significant linguistic investigation of the language (Michaud, 2008; Michaud, 2016).

In this paper we address two research questions. First, is the quality of CLWEs dependent on having large amounts of data in multiple languages, or can large amounts of data in a single *source* lan-

guage inform embeddings trained with little *target* language data? Secondly, can such CLWEs improve language modeling in low-resource contexts by initializing the parameters of an NNLM?

To answer these questions, we scale down the available monolingual data of the target language to as few as 1k sentences, while maintaining a large source language dataset. We assess intrinsic embedding quality by considering correlation with human judgment on the WordSim353 test set (Finkelstein et al., 2001). We then perform language modeling experiments where we initialize the parameters of a long short-term memory (LSTM) language model for low-resource language model training across a variety of language pairs.

Results indicate that CLWEs remain resilient when target language training data is drastically reduced in a simulated low-resource environment, and that initializing the embedding layer of an NNLM with these CLWEs consistently leads to better performance of the language model. In light of these results, we explore the method’s application to Na, an actual low-resource language with realistic manually created lexicons and transcribed data. We present a discussion of the negative results found which highlights challenges and future opportunities.

2 Related Work

This paper draws on work in three general areas, which we briefly describe in this section.

Neural network language models and word embeddings Bengio et al. (2003) and Goodman (2001) introduce word embeddings in the context of an investigation of neural language modeling. One claimed advantage of such models is the ability to cope with sparse data by sharing information among words with similar characteristics. Neural language modeling has since demonstrated powerful capabilities at the word level (Mikolov et al., 2010) and character level (Sutskever et al., 2011). Notably, LSTM models (Hochreiter and Schmidhuber, 1997) for modeling long-ranging statistical influences have been shown to be effective (Graves, 2013; Zaremba et al., 2014).

Word embeddings have become more popular through the application of shallow neural network architectures that allow for training on large quantities of data (Mnih et al., 2009; Bengio et al., 2009; Collobert and Weston, 2008; Mikolov et

al., 2013a), leading to many further investigations (Chen et al., 2013; Pennington et al., 2014; Shazeer et al., 2016; Bhatia et al., 2016). A key application of word embeddings has been in the initializing of neural network architectures for a wide variety of NLP tasks with limited annotated data (Frome et al., 2013; Zhang et al., 2014; Zoph et al., 2016; Lau and Baldwin, 2016).

Low-resource language modeling and language model adaptation Bellegarda (2004) review language model adaptation, and argue that small amounts of in-domain data are often more valuable than large amounts of out-of-domain data, but that adapting background models using in-domain data can be even better. Kurimo et al. (2016) present more recent work on improving large vocabulary continuous speech recognition using language model adaptation for low-resource Finno-Ugric languages.

Cross-lingual language modeling has also been explored with work on interpolation of a sparse language model with one trained on a large amount of translated data (Jensson et al., 2008), and integrated speech recognition and translation (Jensson et al., 2009; Xu and Fung, 2013).

Gandhe et al. (2014) investigate NNLMs for low-resource languages, comparing NNLMs with count-based language models, and find that NNLMs interpolated with count-based methods outperform standard n-gram models even with small quantities of training data. In contrast, our contribution is an investigation into harnessing CLWEs learnt using bilingual dictionaries in order to improve language modeling in a similar low-resource setting.

Cross-lingual word embeddings Cross-lingual word embeddings have also been the subject of significant investigation. Many methods require parallel corpora or comparable corpora to connect the languages (Klementiev et al., 2012; Zou et al., 2013; Hermann and Blunsom, 2013; Chandar A P et al., 2014; Kočiský et al., 2014; Coulmance et al., 2015; Wang et al., 2016), while others use bilingual dictionaries (Mikolov et al., 2013b; Xiao and Guo, 2014; Faruqui and Dyer, 2014; Gouws and Sjøgaard, 2015; Duong et al., 2016; Ammar et al., 2016), or neither (Miceli Barone, 2016).

In particular, we build on the work of Duong et al. (2016). Their method harnesses monolingual corpora in two languages along with a bilingual

lexicon to connect the languages and represent the words in a common vector space. The model builds on the continuous bag-of-words (CBOW) model (Mikolov et al., 2013a) which learns embeddings by predicting words given their contexts. The key difference is that the word to be predicted is a target language translation of a source language word centered in a source language context.

Since dictionaries tend to include a number of translations for words, the model uses an iterative expectation-maximization style training algorithm in order to best select translations given the context. This process thus allows for polysemy to be addressed which is desirable given the polysemous nature of bilingual dictionaries.

3 Resilience of Cross-Lingual Word Embeddings

Previous work using CLWEs typically assumes a similar amount of training data of each available language, often in the form of parallel corpora. Recent work has shown that monolingual corpora of two different languages can be tied together with bilingual dictionaries in order to learn embeddings for words in both languages in a common vector space (Gouws and Søgaard, 2015; Duong et al., 2016). In this section we relax the assumption of the availability of large monolingual corpora on the source and target sides, and report an experiment on the resilience of such CLWEs when data is scarce in the target language but plentiful in a source language.

3.1 Experimental Setup

Word embedding quality is commonly assessed by evaluating the correlation of the cosine similarity of the embeddings with human judgements of word similarity. Here we follow the same evaluation procedure, except where we simulate a low-resource language by reducing the availability of target English monolingual text while preserving a large quantity of source language text from other languages. This allows us to evaluate the CLWEs intrinsically using the WordSim353 task (Finkelstein et al., 2001) before progressing to downstream language modeling where we additionally consider other target languages.

We trained a variety of embeddings on English Wikipedia data of between 1k and 128k sentences from the training data of Al-Rfou et al. (2013). In terms of transcribed speech data, this roughly

equates to between 1 and 128 hours of speech. For the training data, we randomly chose sentences that include words in the WordSim353 task proportionally to their frequency in the set. As monolingual baselines, we use the *skip-gram* (SG) and CBOW methods of Mikolov et al. (2013a) as implemented in the *Gensim* package (Řehůřek and Sojka, 2010). We additionally used off-the-shelf CBOW Google News Corpus embeddings with 300 dimensions, trained on 100 billion words.

The CLWEs were trained using the method of Duong et al. (2016) since their method addresses polysemy which is rampant in dictionaries. The same 1k-128k sentence English Wikipedia data was used but with an additional 5 million sentences of Wikipedia data in a source language. The source languages include Japanese, German, Russian, Finnish, and Spanish, which represent languages of varying similarity with English, some with great morphological and syntactic differences. To relate the languages, we used the *PanLex* lexicon (Kamholz et al., 2014). Following Duong et al. (2016), we used the default window size of 48 so that the whole sentence’s context is almost always taken into account. This mitigates the effect of word re-ordering between languages. We trained with an embedding dimension of 200 for all data sizes as a larger dimension turned out to be helpful in capturing information from the source side.¹

3.2 Results

Figure 1 shows correlations with human judgment in the WordSim353 task. The x-axis represents the number of English training sentences. Coloured lines represent CLWEs trained on different languages: Japanese, German, Spanish, Russian and Finnish.²

With around 128k sentences of training data, most methods perform quite well, with German being the best performing. Interestingly the CLWE methods all outperform GNC which was trained on a far larger corpus of 100 billion words. With only 1k sentences of target training data, all the CLWEs have a correlation around 0.5, with the exception of Finnish. Interestingly, no consis-

¹Hyperparameters for both mono and cross-lingual word embeddings: iters=15, negative=25, size=200, window=48, otherwise default. Smaller window sizes led to similar results for monolingual methods.

²We also tried Italian, Dutch, German and Serbian, yielding similar results but omitted for presentation.

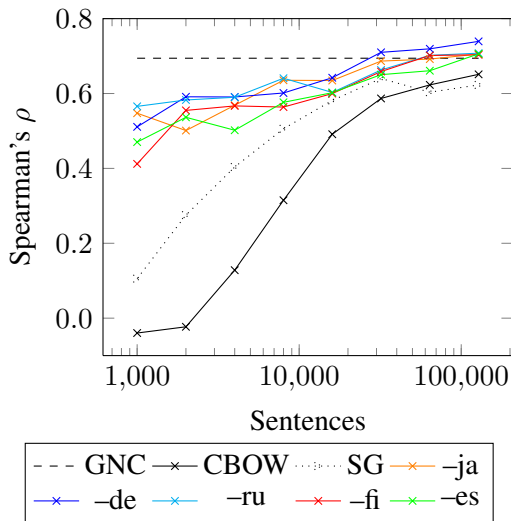


Figure 1: Performance of different embeddings on the WordSim353 task with different amounts of training data. *GNC* is the Google News Corpus embeddings, which are constant. *CBOw* and *SG* are the monolingual word2vec embeddings. The other, colored, lines are all cross-lingual word embeddings harnessing the information of 5m sentences of various source languages.

tent benefit was gained by using source languages for which translation with English is simpler. For example, Spanish often under-performed Russian and Japanese as a source language, as well as the morphologically-rich Finnish.

Notably, all the CLWEs perform far better than their monolingual counterparts on small amounts of data. This resilience of the target English word embeddings suggests that CLWEs can serve as a method of transferring semantic information from resource-rich languages to the resource-poor, even when the languages are quite different. However, the WordSim353 task is a constrained environment, so in the next section we turn to language modeling, a natural language processing task of much practical importance for resource-poor languages.

4 Pre-training Language Models

Language models are an important tool with particular application to machine translation and speech recognition. For resource-poor languages and unwritten languages, language models are also a significant bottleneck for such technologies as they rely on large quantities of data. In this section, we assess the performance of language models on varying quantities of data, across a number

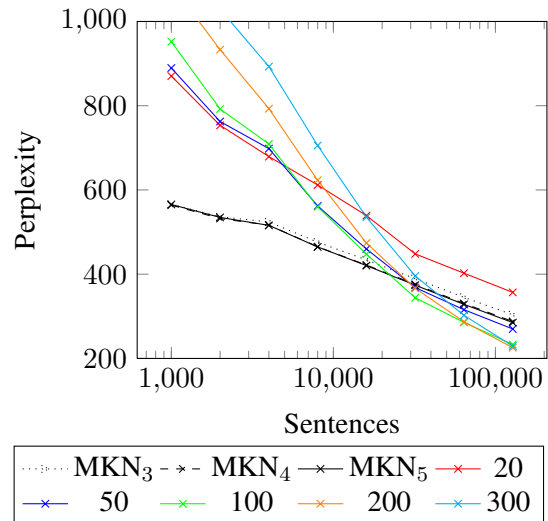


Figure 2: Perplexity of language models on the validation set. Numbers in the legend indicate LSTM language models with different hidden layer sizes, as opposed to Modified Kneser-Ney language models of order 3, 4 and 5.

of different source–target language pairs. In particular, we use CLWEs to initialize the first layer in an LSTM recurrent neural network language model and assess how this affects language model performance. This is an interesting task not simply for the practical advantage of having better language models for low-resource languages. Language modeling is a syntax-oriented task, yet syntax varies greatly between the languages we train the CLWEs on. This experiment thus yields some additional information about how effectively bilingual information can be used for the task of language modeling.

4.1 Experimental Setup

We experiment with a similar data setup as in Section 3. However, target training sentences are not constrained to include words observed in the WordSim353 set, and are random sentences from the aforementioned 5 million sentence corpus. For each language, the validation and test sets consist of 3k randomly selected sentences. The large vocabulary of Wikipedia and the small amounts of training data used make this a particularly challenging language modeling task.

For our NNLMs, we use the LSTM language model of Zaremba et al. (2014). As a count-based baseline, we use Modified Kneser-Ney (MKN) (Kneser and Ney, 1995; Chen and Goodman, 1999) as implemented in KenLM (Heafield,

2011). Figure 2 presents some results of tuning the dimensions of the hidden layer in the LSTM with respect to perplexity on the validation set,³ as well as tuning the order of n-grams used by the MKN language model. A dimension of 100 yielded a good compromise between the smaller and larger training data sizes, while an order 5 MKN model performed slightly better than its lower-order brethren.⁴

Interestingly, MKN strongly outperforms the LSTM on low quantities of data, with the LSTM language model not reaching parity until between 16k and 32k sentences of data. This is consistent with the results of Chen et al. (2015) and Neubig and Dyer (2016) that show that n-gram models are typically better for rare words, and here our vocabulary is large but training data small since the data are random Wikipedia sentences. However these findings are inconsistent with the belief that NNLMs have the ability to cope well with sparse data conditions because of the smooth distributions that arise from using dense vector representations of words (Bengio et al., 2003). Traditional smoothing stands strong.

4.2 English Results

With the parameters tuned on the English validation set as above, we evaluated the LSTM language model when the embedding layer is initialized with various monolingual and cross-lingual word embeddings. Figure 3 compares the performance of a number of language models on the test set. In every case where pre-trained embeddings were used, the embedding layer was held fixed during training. However, we observed similar results when allowing them to deviate from their initial state. For the CLWEs, the same language set was used as in Section 3. The curves for the source languages (Dutch, Greek, Finnish, and Japanese) are remarkably similar, as were those for the languages omitted from the figure (German, Russian, Serbian, Italian, and Spanish). This suggests that the English target embeddings are gleaned similar information from each of the languages, information likely to be more semantic than syntactic, given the syntactic differences between the languages.

³We used 1 hidden layer but otherwise the same as the *SmallConfig* of `models/rnn/ptb/ptb_word_lm.py` available in Tensorflow.

⁴Note that all perplexities in this paper include out-of-vocabulary words, of which there are many.

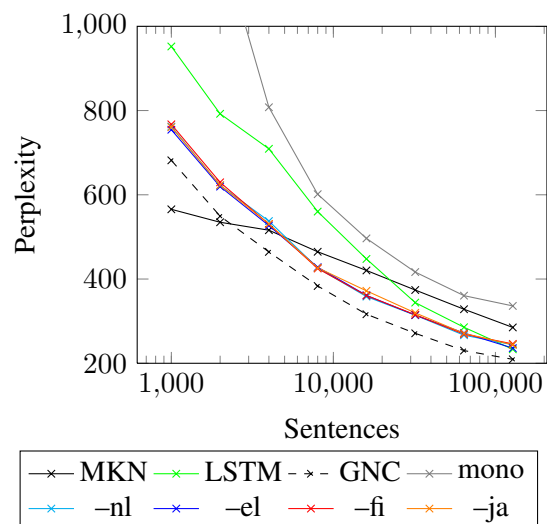


Figure 3: Perplexity of LSTMs when pre-trained with cross-lingual word embeddings trained on the same data. *LSTM* is a neural network language model with no pre-trained embeddings. *mono* is pre-trained with monolingual word2vec embeddings. *GNC* is pre-trained with Google News Corpus embeddings of dimension 300. The rest are pre-trained with CLWEs using information transfer from different source languages. *MKN* is an order 5 Modified Kneser-Ney baseline.

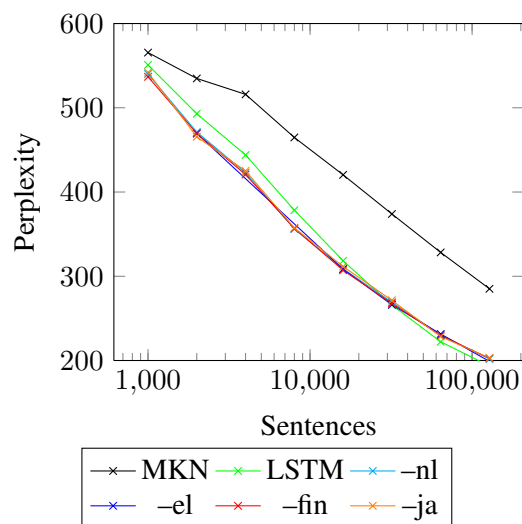


Figure 4: Perplexities when interpolating MKN with LSTMs pre-trained with various cross-lingual word embeddings. *LSTM* interpolates MKN with a neural network language model with no pre-trained embeddings. The rest are interpolations of MKN with LSTMs pre-trained with CLWEs using information transfer from different source languages. *MKN* is an order 5 Modified Kneser-Ney baseline without interpolation.

We compare these language models pre-trained with CLWEs with pre-training using other embeddings. Pre-training with the Google News Corpus embeddings of the method of Mikolov et al. (2013c) unsurprisingly performs the best due to the large amount of English data not available to the other methods, making it a sort of oracle. Monolingual pre-training of word embeddings on the same English data (*mono*) used by the CLWEs yields poorer performance.

The language models initialized with pre-trained CLWEs are significantly better than their un-pre-trained counterpart on small amounts of data, reaching par performance with *MKN* at somewhere just past 4k sentences of training data. In contrast, it takes more than 16k sentences of training data before the plain LSTM language model began to outperform *MKN*. The out-performance of LSTMs by *MKN* with the lowest amounts of training data motivated interpolation of *MKN* probabilities with LSTM language model probabilities, as shown in Figure 4. Such interpolation allows for consistent improvement beyond the performance of *MKN* or CLWE-pre-trained LSTMs alone.

4.3 Other Target Languages

In Table 1 we present results of language model experiments run with other languages used as the low-resource target. In this table English is used in each case as the large source language with which to help train the CLWEs. The observation that the CLWE-pre-trained language model tended to perform best relative to alternatives at around 8k or 16k sentences in the English case prompted us to choose these slices of data when assessing other languages as targets.

The pre-trained LSTM language model outperforms its non-pre-trained counterpart for all languages. There is competition between *MKN* and the CLWE-pre-trained models. The languages for which *MKN* tends to do better are typically those further from English or those with rich morphology, making cross-lingual transfer of information more challenging. There seems to be a degree of asymmetry here: while all languages helped English language modeling similarly, English helps the other languages to varying degrees. For all languages, interpolating *MKN* with the CLWE (*Interp.*) yields the best performance, corroborating the findings of Gandhe et al. (2014).

Neural language modeling of sparse data can be improved by initializing parameters with cross-lingual word embeddings. The consistent performance improvements gained by an LSTM using CLWE-initialization is a promising sign for CLWE-initialization of neural networks for other tasks given limited target language data.

5 First Steps in an Under-Resourced Language

Having demonstrated the effectiveness of CLWE-pre-training of language models using simulation in a variety of well-resourced written languages, we proceed to a preliminary investigation of this method to a low-resource, unwritten language, Na.

Yongning Na is a Sino-Tibetan language spoken by approximately 40k people in an area in Yunnan, China, near the border with Sichuan. It has no orthography and is tonal with a rich morphophonology. Given the small quantity of manually transcribed phonemic data available in the language, Na provides an ideal test bed for investigating the potential and difficulties this method faces in a realistic setting. In this section we report results in Na language modeling and discuss hurdles to be overcome.

5.1 Experimental Setup

The phonemically transcribed corpus⁵ consists of 3,039 phonemically transcribed sentences which are a subset of a larger spoken corpus. These sentences are segmented at the level of the word, morpheme and phonological process, and have been translated into French, with smaller amounts translated into Chinese and English. The corpus also includes word-level glosses in French and English. The lexicon of Michaud (2016) contains example sentences for entries, as well as translations into French, English and Chinese.

The lexicon consists of around 2k Na entries, with example sentences and translations into English, French and Chinese. To choose an appropriate segmentation of the corpus, we used a hierarchical segmentation method where words were queried in the lexicon. If a given word was present then it was kept as a token, otherwise the word was split into its constituent morphemes.

We took 2,039 sentences to be used as training data, with the remaining 1k sentences split

⁵Available as part of the Pangloss collection at <http://lacito.vjf.cnrs.fr/pangloss>.

Lang	8k sentences				16k sentences			
	MKN	LSTM	CLWE	Interp.	MKN	LSTM	CLWE	Interp.
Greek	827.3	920.3	780.4	650.6	749.8	687.9	634.4	549.5
Serbian	492.8	586.3	521.3	408.0	468.8	485.3	447.8	365.7
Russian	1656.8	2054.5	1920.4	1466.2	1609.5	1757.3	1648.3	1309.1
Italian	777.0	794.9	688.3	592.2	686.2	627.7	559.7	493.4
German	997.4	1026.0	1000.9	831.8	980.0	908.8	874.1	761.5
Finnish	1896.4	2438.8	2165.5	1715.3	1963.3	2233.2	2109.9	1641.2
Dutch	492.1	491.3	456.2	381.4	447.9	412.8	378.0	330.1
Japanese	1902.8	2662.4	2475.6	1866.7	1816.8	2462.8	2279.6	1696.9
Spanish	496.3	481.8	445.6	387.7	445.9	412.9	369.6	331.2

Table 1: Perplexity of language models trained on 8k and 16k sentences for different languages. *MKN* is an order 5 Modified Kneser-Ney language model. *LSTM* is a long short-term memory neural network language model with no pre-training. *CLWE* is an LSTM language model pre-trained with cross-lingual word embeddings, using English as the source language. *Interp.* is an interpolation of MKN with CLWE.

	Types	Tokens
Tones	2,045	45,044
No tones	1,192	45,989

Table 2: Counts of types and tokens across the whole Na corpus, given our segmentation method.

	Tones	No tones
MKN	59.4	38.0
LSTM	74.8	46.0
CLWE	76.6	46.2
Lem	76.8	44.7
En-split	76.4	47.0

Table 3: Perplexities on the Na test set using English as the source language. *MKN* is an order 5 Modified Kneser-Ney language model. *LSTM* is a neural network language model without pre-training. *CLWE* is the same LM with pre-trained Na-English CLWEs. *Lem* is the same as CLWE except with English lemmatization. *En-split* extends this by preprocessing the dictionary such that entries with multiple English words are converted to multiple entries of one English word.

equally between validation and test sets. The phonemic transcriptions include tones, so we created two preprocessed versions of the corpus: with and without tones. Table 2 exhibits type and token counts for these two variations. In addition to the CLWE approach used in Sections 3 and 4, we additionally tried lemmatizing the English Wikipedia corpus so that it each token was more likely to be present in the Na-English lexicon.

5.2 Results and Discussion

Table 3 shows the Na language modeling results. Pre-trained CLWEs do not significantly outperform that of the non-pre-trained, and *MKN* outperforms both. Given the size of the training data, and the results of Section 4, it is no surprise that *MKN* outperforms the NNLM approaches. But the lack of benefit in CLWE-pre-training the NNLMs requires some reflection. We now proceed to discuss the challenges of this data to explore why the positive results of language model pre-training that were seen in Section 4 were not seen in this experiment.

Tones A key challenge arises because of Na’s tonal system. Na has rich tonal morphology. Syntactic relationships between words influence the surface form tone a syllable takes. Thus, semantically identical words may take different surface tones than is present in the relevant lexical entry, resulting in mismatches with the lexicon.

If tones are left present, the percentage of Na tokens present in the lexicon is 62%. Removing tones yields a higher hit rate of 88% and allows tone mismatches between surface forms and lexical entries to be overcome. This benefit is gained in exchange for higher polysemy, with an average of 4.1 English translations per Na entry when tones are removed, as opposed to 1.9 when tones are present. Though this situation of polysemy is what the method of Duong et al. (2016) is designed to address, it means the language model fails to model tones and doesn’t significantly help CLWE-pre-training in any case. Future work should investigate morphophonological

processing for Na, since there is regularity behind these tonal changes (Michaud, 2008) which could mitigate these issues if addressed.

Polysemy We considered the polysemy of the tokens of other languages’ corpora in the PanLex dictionaries. Interestingly they were higher than the Na lexicon with tones removed, ranging from 2.7 for Greek–English to 19.5 for German–English. It seems the more important factor is the amount of tokens in the English corpus that were present in the lexicon. For the Na–English lexicon, this was only 18% and 20% when lemmatized and unlemmatized, respectively. However it was 67% for the PanLex lexicon. Low lexicon hit rates of both the Na and English corpora must damage the CLWEs modeling capacity.

Lexicon word forms Not all the forms of many English word groups are represented. For example, only the infinitive ‘*to_run*’ is present, while ‘*running*’, ‘*ran*’ and ‘*runs*’ are not. The limited scope of this lexicon motivates lemmatization on the English side as a normalization step, which may be of some benefit (see Table 3). Furthermore, such lemmatization can be expected to reduce the syntactic information present in embeddings which does not transfer between languages as effectively as semantics.

Some common words, such as ‘*reading*’ are not present in the lexicon, but ‘*to_read_aloud*’ is. Additionally, there are frequently entries such as ‘*way_over_there*’ and ‘*masculine_given_name*’ that are challenging to process. As an attempt to mitigate this issue, we segmented such English entries, creating multiple Na–English entries for each. However, results in Table 3 show that this failed to show improvements. More sophisticated processing of the lexicon is required.

Lexicon size There are about 2,115 Na entries in the lexicon and 2,947 Na–English entries, which makes the lexicon especially small in comparison to the PanLex lexicon used in the previous experiments. Duong et al. (2016) report large reductions in performance of CLWEs on some tasks when lexicon size is scaled down to 10k.

To better understand how limited lexicon size could be affecting language model performance, we performed an ablation experiment where random entries in the PanLex English–German lexicon were removed in order to restrict its size. Figure 5 shows the performance of English language

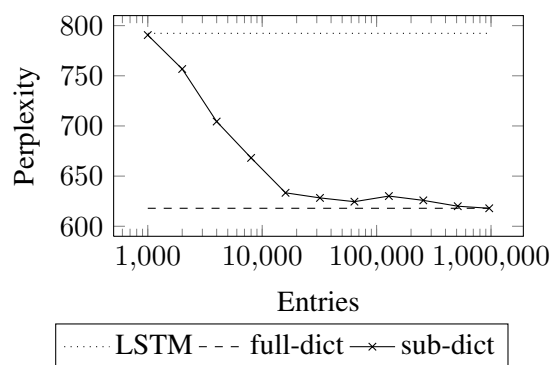


Figure 5: Perplexities of an English–German CLWE-pretrained language model trained on 2k English sentences as the dictionary size available in CLWE training increases to its full size (*sub-dict*). As points of comparison, *LSTM* is a long short-term memory language model with no pre-training and *full-dict* is a CLWE-pretrained language model with the full dictionary available.

modeling when training data is restricted to 2k sentences (to emulate the Na case) and the size of the lexicon afforded to the CLWE training is adjusted. This can only serve as a rough comparison, since PanLex is large and so a 1k entry subset may contain many obscure terms and few useful ones. Nevertheless, results suggest that a critical point occurs somewhere in the order of 10k entries. However, since improvements are demonstrated even with smaller dictionaries, this is further evidence that more sophisticated preprocessing of the Na lexicon is required.

Domain Another difference that may contribute to the results is that the domain of the text is significantly different. The Na corpus is a collection of spoken narratives transcribed, while the Wikipedia articles are encyclopaedic entries, which makes the registers very different.

5.3 Future Work on Na Language Modeling

Though the technique doesn’t work out of the box, this sets a difficult and compelling challenge of harnessing the available Na data more effectively.

The lexicon is a rich source of other information, including part-of-speech tags, example sentences and multilingual translations. In addition to better preprocessing of the lexical information we have already used, harnessing this additional information is an important next step to improving Na language modeling. The corpus includes translations into French, Chinese and English, as well

as glosses. Some CLWE methods can additionally utilize such parallel data (Coulmance et al., 2015; Ammar et al., 2016) and we leave to future work incorporation of this information as well.

The tonal system is well described (Michaud, 2008), and so further Na-specific work should allow differences between surface form tones and tones in the lexicon to be bridged.

Our work corroborates the observation that MKN performs well on rare words (Chen et al., 2015). Interpolation is an effective means to harness this strength when training data is sparse. Furthermore, hybrid count-based and NNLMs (Neubig and Dyer, 2016) promise the best of both worlds for language modeling for low-resource languages.

6 Conclusion

In this paper we have demonstrated that CLWEs can remain resilient when training data in the target language is scaled down drastically. Such CLWEs continue to perform well on the WordSim353 task, as well as demonstrating downstream efficacy across a number of languages through initialization of NNLMs. This work supports CLWEs as a method of transfer of information to resource-poor languages by harnessing distributional information in a large source language. We can expect parameter initialization with CLWEs trained on such asymmetric data conditions to aid in other NLP tasks too, though this should be empirically assessed.

Acknowledgements

This work was conducted during Oliver Adams' internship at IBM Research Australia. We are grateful for support from NSF Award 1464553 and the DARPA/I2O, Contract Nos. HR0011-15-C-0114 and HR0011-15-C-0115.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *arXiv:1602.01925*.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: Review and perspectives. *Speech Communication*, 42(1):93–108.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 490–500.

Steven Bird. 2011. Bootstrapping the language archive: New prospects for natural language processing in preserving linguistic heritage. *Linguistic Issues in Language Technology*, 6:1–16.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mm Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems 27*, pages 1853–1861.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Yanqing Chen, Bryan Perozzi, R Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv:1301.3226*.

Welin Chen, David Grangier, and Michael Auli. 2015. Strategies for training large vocabulary neural language models. *arXiv:1512.04906*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th international conference on Machine Learning*, pages 160–167.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Transgram, fast cross-lingual word-embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1113.

Thi-Ngoc-Diep Do, Alexis Michaud, and Eric Castelli. 2014. Towards the automatic processing of Yongning Na (Sino-Tibetan): developing a ‘light’ acoustic model of the target language and testing ‘heavy-weight’ models from five national languages. In *4th International Workshop on Spoken Language Technologies for Under-resourced Languages*, pages 153–160.

- Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1285–1295.
- Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppín. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web, WWW '01*, pages 406–414.
- Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc’Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems 26*, pages 2121–2129.
- Ankur Gandhe, Florian Metz, and Ian Lane. 2014. Neural network language models for low resource languages. In *INTERSPEECH-2014*, pages 2615–2619.
- Joshua Goodman. 2001. A Bit of Progress in Language Modeling. *Technical Report*.
- Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390.
- Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv:1308.0850*.
- Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.
- Karl Moritz Hermann and Phil Blunsom. 2013. A simple model for learning multilingual compositional semantics. *arXiv:1312.6173*.
- Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and Others. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Arnar Jensson, Koji Iwano, and Sadaoki Furui. 2008. Development of a speech recognition system for icelandic using machine translated text. In *The first International Workshop on Spoken Languages Technologies for Under-resourced Languages*, pages 18–21.
- Arnar Jensson, Tasuku Oonishi, Koji Iwano, and Sadaoki Furui. 2009. Development of a WFST based speech recognition system for a resource deficient language using machine translation. *Proceedings of Asia-Pacific Signal and Information Processing Association*, pages 50–56.
- David Kamholz, Jonathan Pool, and Susan Colowick. 2014. PanLex: Building a resource for panlingual lexical translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 3145–3150.
- Alexandre Klementiev, Ivan Titov, and Binod Bhat-tarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474.
- Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.
- Tomáš Kočiský, Karl Moritz Hermann, and Phil Blun-som. 2014. Learning bilingual word representations by marginalizing alignments. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 224–229.
- Mikko Kurimo, Seppo Enarvi, Ottokar Tilk, Matti Var-jokallio, André Mansikkaniemi, and Tanel Alumäe. 2016. Modeling under-resourced languages for speech recognition. *Language Resources and Evaluation*, pages 1–27.
- Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *1st Workshop on Representation Learning for NLP*, pages 78–86.
- Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126.
- Alexis Michaud. 2008. Phonemic and tonal analysis of Yongning Na*. *Cahiers de Linguistique Asie Orientale*, 37(2):159–196.
- Alexis Michaud. 2016. Online Na-English-Chinese Dictionary. <https://halshs.archives-ouvertes.fr/halshs-01204638>. This is version 1.1 of the dictionary.
- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH-2010*, pages 1045–1048.

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representation in vector space. *arXiv:1301.3781*.
- Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv:1309.4168*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Andriy Mnih, Zhang Yuecheng, and Geoffrey Hinton. 2009. Improving a statistical language model through non-linear prediction. *Neurocomputing*, 72(7-9):1414–1418.
- Graham Neubig and Chris Dyer. 2016. Generalizing and hybridizing count-based and neural language models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1163–1172.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.
- Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.
- Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving embeddings by noticing what’s missing. *arXiv:1602.02215*.
- Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1017–1024.
- Rui Wang, Hai Zhao, Sabine Ploux, Bao-Liang Lu, and Masao Utiyama. 2016. A novel bilingual word embedding method for lexical translation using bilingual sense clique. *arXiv:1607.08692*.
- Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129.
- Ping Xu and Pascale Fung. 2013. Cross-lingual language modeling for low-resource speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 21(6):1134–1144.
- Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv:1409.2329*.
- Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 111–121.
- Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv:1604.02201*.
- Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Consistent Translation of Repeated Nouns using Syntactic and Semantic Cues

Xiao Pu^{1,2}, Laura Mascarell³ and Andrei Popescu-Belis¹

¹Idiap Research Institute, 1920 Martigny, CH

²École Polytechnique Fédérale de Lausanne, 1015 Lausanne, CH

³Institute of Computational Linguistics, University of Zürich, 8050 Zürich, CH

{xiao.pu, andrei.popescu-belis}@idiap.ch

{mascarell}@cl.uzh.ch

Abstract

We propose a method to decide whether two occurrences of the same noun in a source text should be translated consistently, i.e. using the same noun in the target text as well. We train and test classifiers that predict consistent translations based on lexical, syntactic, and semantic features. We first evaluate the accuracy of our classifiers intrinsically, in terms of the accuracy of consistency predictions, over a subset of the UN Corpus. Then, we also evaluate them in combination with phrase-based statistical MT systems for Chinese-to-English and German-to-English. We compare the automatic post-editing of noun translations with the re-ranking of the translation hypotheses based on the classifiers' output, and also use these methods in combination. This improves over the baseline and closes up to 50% of the gap in BLEU scores between the baseline and an oracle classifier.

1 Introduction

The repetition of a noun in a text may be due to co-reference, i.e. repeated mentions of the same entity, or to mentions of two entities of the same type. But in other cases, two occurrences of the same noun may simply convey different meanings. The translation of repeated nouns depends, among other things, on the conveyed meanings: in case of co-reference or identical senses, they should likely be translated with the same word, while otherwise they should be translated with different words, if the target language distinguishes the two meanings. State-of-the-art machine translation systems do not address this challenge systematically, and translate two occurrences of the same noun inde-

pendently, thus potentially introducing unwanted variations in translation.

We exemplify this issue in Figure 1 for Chinese-to-English and German-to-English translations, with examples of inconsistent translations of a repeated source noun by a baseline SMT system, as opposed to consistent translations in the reference. In Example 1, the system's translation of the second occurrence of *politik* is mistaken and should be replaced by the first one (*policy*, not *politics*). In Example 2, although the first translation differs from the reference, it could be acceptable as a legitimate variation, although the second one (*identity documents*) is more idiomatic and more frequent. Of course, in addition to these two examples, there are other configurations of the six nouns involved in a consistency relation across source, candidate and reference translations, but they will be discussed below when designing the training and test data for our problem.

In this paper, we aim to improve the translation of repeated nouns by designing a classifier which predicts, for every pair of repeated nouns in a source text, whether they should be translated by the same noun, i.e. consistently, and if that is the case, which of the two candidate translations generated by an MT system should replace the other one. We thus address one kind of long-range dependencies between words in texts; such dependencies have been the target of an increasing number of studies, presented briefly in Section 2.

To learn a consistency classifier from the data, we consider a corpus with source texts and reference translations, from the parallel UN Corpora in Chinese, German and English. As we explain in Section 3, we mine the corpus for pairs of repeated nouns in the source texts, and examine human and machine translations in order to learn to predict whether the machine translation of the first noun must replace the second one, or vice-versa, or no

Example 1

Source: nach einföhrung dieser politik [...] die politik auf dem gebiet der informationstechnik [...]

Reference: once the policy is implemented [...] the information technology policy [...]

MT: after introduction of policy [...] the politics in the area of information technology [...]

Example 2

Source: 欺詐性旅行或身份證件系指有下列情形之一的任何旅行或身份證件

Reference: Fraudulent travel or identity document; shall mean any travel or identity document

MT: 欺詐性 travel or identity papers. 系指 have under one condition; any travel, or identity document

Figure 1: Inconsistent translations of repeated nouns, in blue, from German (Example 1) and Chinese (Example 2) into English. While in both examples one noun is different from the reference, only Example 1 is truly mistaken: the second occurrence of the noun should be replaced with the first one.

change should be made. In Section 4, we present the lexical, syntactic and semantic features used by the classifiers. When presented with previously unseen source texts and baseline MT output, the decisions of the classifiers serve to post-edit or re-rank the repeated nouns of the MT baseline.

As shown in Section 5, the new end-to-end MT system generates improved Chinese-English and German-English translations, with larger improvements on the latter pair. Syntactic features appear to be more useful than semantic ones, for reasons that will be discussed. The case of more than two consecutive occurrences of the same noun will be briefly examined. Finally, a combined re-ranking and post-editing approach appears to be the most effective, covering about 50% of the gap in BLEU scores between the baseline MT and the use of an oracle classifier.

2 Related Work

This study is related to several research topics in MT: lexical consistency, caching, co-reference, and long-range dependencies between words in general. Our proposal aims to improve the consistency of noun translation, and thus has a narrower scope than the “one translation per discourse” hypothesis (Carpuat, 2009; Carpuat and Simard, 2012), which aimed to implement for MT the broader hypothesis of “one sense per discourse” (Gale et al., 1992).

We focus on nouns because of their referential properties, which are a strong requirement for consistency in case of co-reference, although in many cases consistency should not be blindly enforced, in order to avoid the “trouble with MT consistency” (Carpuat and Simard, 2012) which may induce translation errors. As indicated in that study, MT systems trained on small datasets are often

more consistent but of lower quality than systems trained on larger and more diverse data sets. In any case, in our study, we never alter consistent translations, but we address inconsistencies, which are often translation errors (Carpuat and Simard, 2012), and attempt to find those that can be corrected simply by enforcing consistency.

Similarly, our scope is narrower than the caching approach (Tiedemann, 2010; Gong et al., 2011), which encourages *a priori* consistent translations of any word, with the risk on propagating cached incorrect translations. In our study, the first and second translation in a pair have equal status.

Noun phrase consistency is often due to co-reference. Several recent studies consider co-reference to improve pronoun resolution, but none of them exploits noun phrase co-reference, likely due to an insufficient accuracy of co-reference resolution systems (?; ?). The improvement of pronoun translation was only marginal with respect to a baseline SMT system in a 2015 shared task (Hardmeier et al., 2015), while the 2016 shared task (Guillou et al., 2016) somewhat shifted its focus to pronoun prediction in a lemmatized reference translation.

This study builds upon and extends our previous work on the translation of compounds (Mascarell et al., 2014; Pu et al., 2015), which constrained the translation of the head of a compound when it was repeated separately after it. The present study is considerably more general, as it makes no assumption on either of the repeated nouns, i.e. it does not require them to be part of compounds.

Our study contributes to a growing corpus of research on modeling longer-range dependencies than those modeled in phrase-based SMT or neural MT, often across different sentences of a document. Ture et al. (2012) used cross-sentence

consistency features in a translation model, while Hardmeier (2012) designed the Docent decoder, which can use document-level features to improve the coherence across sentences of a translated document. Our classifier for repeated nouns outputs decisions that can serve as features in Docent, but as the frequency of repeated nouns in documents is quite low, we use here post-editing and/or re-ranking rather than Docent.

3 Datasets for Noun Consistency in MT

3.1 Overview of the Method

Our method flexibly enforces noun consistency in discourse to improve noun phrase translation. We first detect two neighboring occurrences of the same noun in the source text, i.e. closer than a fixed distance, and which satisfy some basic conditions. Then, we consider their baseline translations by a phrase-based statistical MT system, which are identified from word-level alignments. If the two baseline translations of the repeated noun differ, then our classifier uses the source and target nouns and a large set of features (presented in Section 4) to decide whether one of the translations should be edited, and how. This decision will serve to post-edit and/or re-rank the baseline MT’s output (Section 4.4). To design the classifier, we train machine-learning classifiers over examples that are extracted from parallel data and from a baseline MT system, as described in Section 3.3. A separate subset of unseen examples will be used to test classifiers, first intrinsically and then in combination with MT.

3.2 Corpora and Pre-processing

Our data comes from WIT³ Corpus¹ (Cettolo et al., 2012), a collection of transcripts of TED talks, and the UN Corpora,² a collection of documents from the United Nations. The experiments are on Chinese-to-English and German-to-English.

We first build a phrase-based SMT system for each language pair with Moses (Koehn et al., 2007), with its default settings. Both MT systems are trained on the WIT³ data, and are used to generate candidate translations of the UN Corpora. Then, the ML classifiers are trained on noun pairs extracted from the UN Corpora, using semantic and syntactic features extracted from both source and target sides. The test sets also come

from the UN Corpora, with the same features on the source side. Table 1 presents statistics about the data.

3.3 Extraction of Training/Testing Instances

At this stage, the goal is to automatically extract for training the pairs of repeated nouns in the source texts, noted $N \dots N$, which are translated differently by the SMT baseline, noted $T_1 \dots T_2$, with $T_1 \neq T_2$. Indeed, when the translations are identical, we have no element in the 1-best translation to post-edit them, therefore we do not consider such pairs. We examine the reference translations of T_1 and T_2 , noted RT_1 and RT_2 , from which we derive the answer we expect from the classifiers (as specified below), and which will be used for supervised learning. We obtain the T_i and RT_i values using word-alignment with GIZA++.

Prior to the identification of repeated nouns in the source text, we tokenize the texts and identify parts-of-speech (POS) using the Stanford NLP tools³. In particular, as Chinese texts are not word-segmented, we first perform this operation and then identify multi-character nouns. We then consider each noun in turn, and look for a second occurrence of the same noun in what follows, limiting the search to the same sentence for Chinese, and to the same and next three sentences for German. The difference in the distance settings is based on observations of the Chinese vs. German datasets: average length of sentences, average distance of repeated nouns, and sentence segmentation issues.

Once the pairs of repeated nouns have been identified, we check the SMT translations of each pair, and if the two translations are different, we include the pair in our dataset. For instance, in Figure 1, the noun 证件 appears twice in the sentence, and the baseline translations of the two occurrences are *papers* and *document*; therefore, this pair is included in our dataset. We extracted from the UN Corpora 3,301 pairs for training and 647 pairs for testing on ZH/EN, and 11,289 pairs for training and 695 pairs for testing on DE/EN. We selected a smaller amount of noun pairs for ZH/EN than DE/EN for reasons of availability, because DE/EN dataset is more than 10 times larger than the ZH/EN one. We kept similar test set sizes to enable comparison.

The word-aligned reference translations are

¹<http://wit3.fbk.eu>

²<http://www.uncorpora.org>

³<http://nlp.stanford.edu/software>

WIT ³	MT training		MT tuning		Language modeling	
	Sentences	Words	Sentences	Words	Sentences	Words
DE-EN	193,152	3.6M	2,052	40K	217K	4.4M
ZH-EN	185,443	3.4M	2,457	54K	4.8M	800M

UN Data	Classifier training			Classifier testing		
	Sentences	Words	Nouns	Sentences	Words	Noun
DE-EN	150K	4.5M	11,289	7,771	225K	695
ZH-EN	10K	368K	3,301	3,000	121K	647

Table 1: WIT³ data for building the SMT systems and UN data to train/test the classifiers.

used to set the ground-truth class (or decision) for training the classifiers, as follows. With the notations above (baseline translations of N noted T_1 and T_2 , with $T_1 \neq T_2$), if the reference translations differ ($RT_1 \neq RT_2$), then we label the pair as ‘none’, i.e. none of T_1 and T_2 should be post-edited and changed into the other, because this would not help to reach the reference translation anyway (recall that the only possible actions knowing the SMT baseline are replacing T_1 by T_2 or vice-versa).

If the reference translations are the same ($RT_1 = RT_2$), then we examine this word, noted RT . If this word is equal to one of the baseline translations ($T_1 = RT$ or $T_2 = RT$), then this value should be given to other baseline (e.g., if $T_1 = RT \neq T_2$, then $T_2 := T_1$). For classification, we simply label these examples with the index of the word that must be used, 1 or 2. However, if the reference differs from both baseline translations, then the label is again ‘none’, because we cannot infer which of them is a better translation.

After labeling all the pairs, we extract the features in an attribute/value format to be used for machine learning.

4 Classifiers for Translation Consistency

4.1 Role and Nature of the Classifiers

We describe here the machine learning classifiers that are trained to predict one of the three classes – ‘1’, ‘2’ or ‘none’ – for each pair of identical source nouns with different baseline SMT translations. The sense of the predicted classes is the following: ‘1’ means that T_1 should replace T_2 , ‘2’ means the opposite, and ‘none’ means translations should be left unchanged. For instance, if Example 2 in Figure 1 was classified as ‘2’, we would replace the translation of the first occurrence (*papers*) with the second one (*documents*).

We use the WEKA environment⁴ to train and test several different learning algorithms: SVMs (Cortes and Vapnik, 1995), C4.5 Decision Trees (noted J48 in Weka) (Quinlan, 1993), and Random Forests (Breiman, 2001). We use 10-fold cross validation on the training set, and then test them once on the test set, and later on in combination with MT. For performance reasons, we used the Maximum Entropy classifier (Manning and Klein, 2003) from Stanford⁵ instead of WEKA’s Logistic Regression.

The hyper-parameters of the above classifiers were set as follows, mostly following the default settings from WEKA, and setting others on the cross-validation sets (not the unseen test sets). For SVMs, the round-off error is $\epsilon = 10^{-12}$. For Decision Trees, we set the minimal number of instances per leaf (‘minNumObj’) at 2 and the confidence factor used for pruning to 0.25. For Random Forests, we defined the number of trees to be generated (‘numTree’) as 100 and set their maximal depth (‘maxDepth’) as unlimited. Finally, we set the MaxEnt smoothing (σ) to 1.0, and the tolerance used for convergence in parameter optimization to 10^{-5} .

We evaluate our proposal in two ways. First, we measure the classification accuracy in terms of *accuracy* and *kappa* (κ) agreement (Cohen, 1960) with the correct class, either in 10-fold cross-validation experiments, or on the test set. Second, we compare the updated translations with the reference, to check if we obtain a result that is closer to it, using the popular BLEU measure (Papineni et al., 2002).

4.2 Syntactic Features

We defined 19 syntactic features, mainly with the assumption that out of a pair of repeated source nouns $N \dots N$, the occurrence which is embed-

⁴<http://www.cs.waikato.ac.nz/ml/weka/>

⁵<http://nlp.stanford.edu/software/classifier.shtml>

ded in a more complex local parse tree, i.e. has more information syntactically bound to it, is more “determined” and has a higher probability of been translated correctly by the baseline MT system, since this information can help the system to disambiguate it. The results tend to confirm this assumption.

The features are listed in Figure 2, left side, with an explicit description of each feature and its value on a Chinese text (top of the figure). In the last line of the table we show the ground-truth class of this example.

The sentences are parsed using the Stanford parser,⁶ and the values of the features are obtained from the parse trees, using the sizes (in nodes or words) of the siblings and ancestor sub-trees for each analyzed noun. In the sample parse trees on the right side of Figure 2, the first NP ancestor is marked with a red rectangle, and the values of the features are computed

We can distinguish three subsets of features. The first subset includes lexical and positional features: the original noun, automatic baseline translations of both occurrences from the baseline MT system, and the distance between the sentences that contain the two nouns. The second subset includes features that capture the size of the siblings in the parse trees of each of the two nouns. The third subset includes the size of sub-tree for the latest noun phrase ancestor for each analyzed noun, and also the depth distances to the next noun phrase ancestor.

4.3 Semantic Features

The semantic features, to be used independently or in combination with the syntactic ones, are divided into two groups: discourse vs. local context features, which differ by the amount of context they take into account. On the one hand, local context features represent the immediate context of each of the nouns in the pair and their translations, i.e. three words to their left and three words to their right in both source and MT output, always within the same sentence.

On the other hand, discourse features capture those cases where the inconsistent translations of a noun might be due to a disambiguation problem of the source noun, and semantic similarity can be leveraged to decide which of the two translations best matches the context. To compute the

discourse features, we use the word2vec word vector representations generated from a large corpus (Mikolov et al., 2013), which have been successfully used in the recent past to compute similarity between words (Schnabel et al., 2015). Specifically, we employ the model trained on the English Google News corpus⁷ with about 100 billion words.

For each pair of inconsistent translations (T_1 , T_2) of a source noun N , we compute the cosine similarities c_1 and c_2 between the vector representation of each translation and the mean vector of their contexts. These mean vectors, noted \vec{v}_1 and \vec{v}_2 , are computed by averaging all vectors of the words in the respective contexts of T_1 and T_2 . Here, the contexts consist of 20 words to the left and 20 words to the right of each T_i , possibly crossing sentence boundaries. The cosine similarities c_1 and c_2 are thus:

$$c_1 = \cos(\vec{T}_1, \vec{v}_1) = \frac{\vec{T}_1 \cdot \vec{v}_1}{\|\vec{T}_1\| \|\vec{v}_1\|} \quad (1)$$

$$c_2 = \cos(\vec{T}_2, \vec{v}_2) = \frac{\vec{T}_2 \cdot \vec{v}_2}{\|\vec{T}_2\| \|\vec{v}_2\|} \quad (2)$$

The two values c_1 and c_2 are used as features, allowing classifiers to learn that, in principle, higher values indicate a better translation in the sense of its semantic similarity with the context.

In the Example 1 from Figure 1, the German word *Politik* is translated into the English words *policy* and then *politics*. The semantic similarity between the word *politics* and its context (c_2) is lower than the similarity between *policy* and its context (c_1), which we consider to be an indication that the first occurrence, namely *policy*, has better chances to be the correct translation – which is actually the case in this example.

4.4 Integration with the MT System

The classifier outputs a post-editing decision for each pair of repeated nouns: replace T_1 with T_2 , replace T_2 with T_1 , or do nothing. This decision can be directly executed, or it can be combined in a more nuanced fashion with the MT system. Therefore, to modify translations using this decision, we propose and test three approaches for using in in an MT system:

Post-editing: directly edit the translations T_1 or T_2 depending on the classifier’s decision.

⁶<http://nlp.stanford.edu/software/lex-parser.html>

⁷<https://code.google.com/p/word2vec/>

Source: 赞扬 联合国 人权 事务 高级 专员 办事处 高度 优先 从事 有关 国家 机构 的工作 , [. . .] , 鼓励 高级 专员 确保 作出 适当 安排 和 提供 预算 资源

Reference: commends the high priority given by the office of the united nations high commissioner for human rights to work on national institutions , [. . .] , encourages the high commissioner to ensure that appropriate arrangements are made and budgetary resources provided

MT: praise the human rights high commissioner was the high priority to offices in the country , [. . .] , to encourage senior specialists to make sure that make appropriate and provide budget resources

Features	Values
Source noun (Chinese)	专员
Distance in sentences between the two source occurrences	0
Translation of the first occurrence (labeled NN)	commissioner
Translation of the second occurrence (labeled NN)	specialists
Number of sibling nodes of the 1 st occurrence	4
Number of sibling nodes of the 2 nd occurrence	2
Sign of the difference between the above (+1, 0, -1)	1
Number of words of the 1 st occurrence and its siblings	2
Number of words of the 2 nd occurrence and its siblings	1
Sign of the difference between the above (+1, 0, -1)	1
Number of nodes in the first NP ancestor of 1 st occ.	15
Number of nodes in the first NP ancestor of 2 nd occ.	7
Sign of the difference between the above (+1, 0, -1)	1
Number of words in the first NP ancestor of the 1 st occ.	6
Number of words in the first NP ancestor of the 2 nd occ.	2
Sign of the difference between the above (+1, 0, -1)	1
Distance between the first NP ancestor and the 1 st occ.	3
Distance between the first NP ancestor and the 2 nd occ.	3
Sign of the difference between the above (+1, 0, -1)	0
Class (1, 2, 0)	1

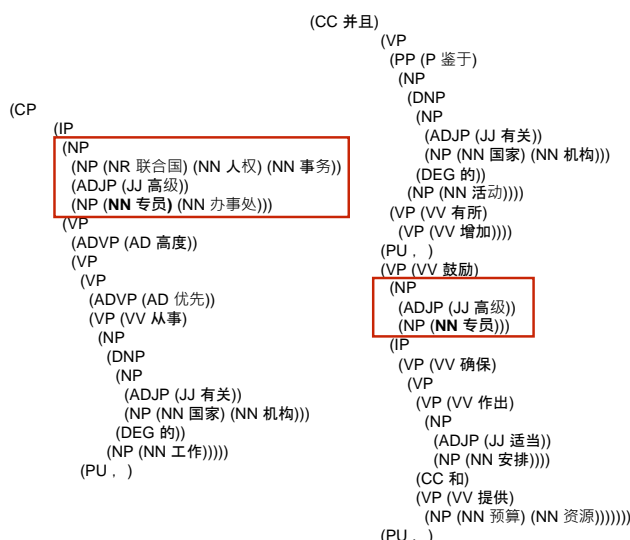


Figure 2: Definition of syntactic features (left) and illustration of their values on a Chinese text (top). The red boxes in the parse trees (right) show the first NP ancestors of the examined nouns.

Re-ranking: search among the translation hypotheses provided by the SMT system (in practice, the first 10,000 ones) for those where T_1 and T_2 are translated as predicted by the classifier, and select the highest ranking one as the new translation. If none is found, the baseline 1-best hypothesis is kept.

Re-ranking + Post-editing: after applying re-ranking, if no hypothesis conforms to the prediction of the classifier, instead of keeping the baseline translation we post-edit it as in the first approach.

5 Results and Analysis

We first present the results of the classification task, i.e. the prediction of the correct translation variant (1st / 2nd / None), for Chinese-English and German-English translation respectively in Tables 2 and 3, with 10-fold cross-validation on the training sets. Then, we present the scores on the test sets for both the classification task and its

	Syntactic features		Semantic features		All features	
	Acc. (%)	κ	Acc. (%)	κ	Acc. (%)	κ
J48	72.1	0.48	60.2	0.00	60.2	0.00
SVM	74.5	0.54	60.2	0.00	73.9	0.51
RF	75.3	0.54	68.4	0.29	70.7	0.35
MaxEnt	76.7	0.65	69.5	0.32	83.3	0.75

Table 2: Prediction of the correct translation (1st / 2nd / None) for repeated nouns in *Chinese*, in terms of accuracy (%) and kappa scores, on the development set with 10-fold cross-validation. Methods are sorted by average accuracy over the three feature sets. When using semantic or all features, no decision tree outperformed the majority class baseline, hence $\kappa = 0$.

combination with MT, for ZH/EN and DE/EN respectively in Tables 4 and 5. We compare the results obtained with several ML methods: Decision Trees (J48), SVMs, Random Forests and MaxEnt, ordered in the tables by average increasing scores. Moreover, we compare the merits of syntactic vs. semantic features, as well as post-editing vs. re-ranking the MT output.

	Syntactic features					Semantic features					All features				
	Acc.	κ	BLEU			Acc.	κ	BLEU			Acc.	κ	BLEU		
			PE	RR	RR+PE			PE	RR	RR+PE			PE	RR	RR+PE
Baseline	-	-	11.07	11.07	11.07	-	-	11.07	11.07	11.07	-	-	11.07	11.07	11.07
J48	66.3	0.42	11.17	11.20	11.30	33.1	0.00	11.07	11.07	11.07	33.1	0.00	11.07	11.07	11.07
SVM	71.9	0.53	11.23	11.27	11.33	33.1	0.00	11.07	11.07	11.07	62.1	0.43	11.18	11.26	11.26
RF	71.7	0.53	11.22	11.24	11.27	55.2	0.33	11.04	11.07	11.12	54.9	0.32	11.16	11.20	11.24
MaxEnt	73.7	0.60	11.27	11.33	11.35	56.1	0.34	10.87	11.11	11.18	72.5	0.56	11.21	11.33	11.36
Oracle	100	1.00	11.40	11.52	11.64	100	1.00	11.40	11.52	11.64	100	1.00	11.40	11.52	11.64

Table 4: Prediction of the correct translation (accuracy (%) and κ) and translation quality (BLEU) for repeated nouns on the *Chinese test set*. Maximum Entropy was the best method found on the dev set.

	Syntactic features					Semantic features					All features				
	Acc.	κ	BLEU			Acc.	κ	BLEU			Acc.	κ	BLEU		
			PE	RR	RR+PE			PE	RR	RR+PE			PE	RR	RR+PE
Baseline	-	-	17.10	17.10	17.10	-	-	17.10	17.10	17.10	-	-	17.10	17.10	17.10
SVM	71.4	0.57	17.59	17.65	17.72	32.8	0.00	17.10	17.10	17.10	32.8	0.00	17.10	17.10	17.10
J48	70.5	0.56	17.59	17.61	17.70	48.2	0.23	17.13	17.27	17.33	69.4	0.54	17.56	17.60	17.66
RF	70.2	0.55	17.55	17.62	17.68	54.4	0.32	17.21	17.34	17.37	67.6	0.52	17.53	17.57	17.63
MaxEnt	78.3	0.67	17.63	17.66	17.75	63.5	0.49	17.39	17.47	17.49	68.7	0.53	17.58	17.59	17.67
Oracle	100	1.00	17.78	17.83	17.99	100	1.00	17.78	17.83	17.99	100	1.00	17.78	17.83	17.99

Table 5: Prediction of the correct translation (accuracy (%) and κ) and translation quality (BLEU) for repeated nouns on the *German test set*. Maximum Entropy was the best method found on the dev set.

	Syntactic features		Semantic features		All features	
	Acc. (%)	κ	Acc. (%)	κ	Acc. (%)	κ
SVM	77.8	0.67	38.1	0.00	38.1	0.00
J48	77.0	0.66	64.8	0.45	79.7	0.69
RF	82.0	0.73	73.5	0.60	84.5	0.77
MaxEnt	80.8	0.71	76.8	0.65	83.4	0.75

Table 3: Prediction of the correct translation (1st / 2nd / None) for repeated nouns in *German*, in terms of accuracy (%) and kappa scores, on the *development set* with 10-fold c.-v. Methods are sorted by average accuracy over the three feature sets. The best scores are in bold.

5.1 Best Scores of Classification and MT

The classification accuracy is above 80% when applying 10-fold cross-validation, for both language pairs, and reaches 74–78% on the test sets. As the classes are quite balanced, a random baseline would reach around 33% only. Kappa values reach 0.75 on the dev sets and 0.60–0.67 on the test sets. The performances of the classifiers appear thus to be well above chance, and the comparable performances achieved on the unseen test sets indicate that over-fitting is unlikely.

The ordering of methods by performance is remarkably stable: Decision Trees (J48) and SVMs get the lowest scores, followed by Random Forests, and then by the MaxEnt classifier. The ordering {J48, SVM} < RF < MaxEnt is observed over both language pairs, over the three types of features, and the four datasets, with 1-2 exceptions only. Overall, the best configuration of our method found on the training sets is, for both languages, the MaxEnt classifier with all features.

There is a visible rank correlation between the increase in classification accuracy and the increase in BLEU score, for all languages, features, classifiers, and combination methods with MT. The best configurations found on the training sets bring the following BLEU improvements: for ZH/EN, from 11.07 to 11.36, and for DE/EN, from 17.10 to 17.67. In fact, syntactic features turn out to reach an even higher value on the test set, at 17.75. To interpret these improvements, they should be compared to the oracle BLEU scores obtained by using a “perfect” classifier, which are 11.64 for ZH/EN and 17.99 for DE/EN. Our method thus bridges 51% of the BLEU gap between baseline and oracle on ZH/EN and 64% on DE/EN – a significant improvement.

The BLEU scores of the three different methods for using classification for MT (Tables 4 and 5) clearly show that the combined method outperforms both post-editing and re-ranking alone, for all languages and features. Post-editing, the easiest one to implement, has little consideration for the words surrounding the nouns, while re-ranking works on MT hypotheses and thus ensures that a better global translation is found that is also consistent. However, in some cases, no hypothesis conforms to the consistency decision, and in this case post-editing the best hypothesis appears to be beneficial.

5.2 Feature Analysis: Syntax vs. Semantics

On the training sets, syntactic features always outperform the semantic ones when using the Max-

ZH/EN		DE/EN	
Translation of the 2 nd occurrence	0.165	Translation of the 1 st occurrence	0.162
Translation of the 1 st occurrence	0.163	Translation of the 2 nd occurrence	0.162
Source noun	0.110	Source noun	0.099
#words in the first NP ancestor of the 2 nd occ.	0.060	#words in the first NP ancestor of the 2 nd occ.	0.062
#words in the first NP ancestor of the 1 st occ.	0.050	#words in the first NP ancestor of the 1 st occ.	0.057
#nodes in the first NP ancestor of the 2 nd occ.	0.036	#nodes of the 2 nd occ.	0.054
#nodes in the first NP ancestor of the 1 st occ.	0.033	#sibling nodes of the 1 st occ.	0.052
Sign of difference between #words and its siblings	0.031	#nodes in the first NP ancestor of the 1 st occ.	0.042
Dist. between the first NP ancestor and the 2 nd occ.	0.025	#nodes in the first NP ancestor of the 2 nd occ.	0.037
#words of the 1 st occ. and its siblings	0.023	#words of the 2 nd occ. and its siblings	0.037

Table 6: Top ten syntactic features ranked by information gain for each language pair.

Ent classifier, and their joint use outperforms their separate uses. For the other classifiers (not the best ones on the training sets), on ZH/EN, adding semantic features to syntactic ones decreases the performance. Indeed, semantic features (specifically the discourse ones) are intended to disambiguate nouns based on contexts, but here, manual inspection of the data showed that these are similar for T_1 and T_2 , which makes prediction difficult.

Semantic features appear to be more useful in German compared to Chinese. We hypothesize that this is because translation ambiguities of Chinese nouns, i.e. cases when the same noun can be translated into English with two very different words, are less frequent and less semantically divergent than in German. In other words, semantic features are less useful in Chinese because cases of strong polysemy or homonymy seem to be less frequent than in German. Such a characteristic is suggested for English vs. Chinese by Huang (1995), and we believe it extends to German.

These facts might also explain the results obtained when using all features, for German and Chinese. As in Chinese semantic features are less helpful, given also the limited amount of data, combining them with syntactic ones actually decreases the performance of the syntactic ones used independently. In contrast, semantic features are more helpful on German dataset, and also improve results when we considered along with the syntactic ones together.

Table 6 shows the top ten syntactic features for ZH/EN and for DE/EN, ranked by information gain computed using Weka. These features include both lexical information and properties of the parse tree. The analysis shows that lexical features are significantly more important than purely syntactic ones, for both languages. However, the syntactic ones are not negligible.

cosSim.	Inst.	Local Context		Discourse		Both	
		Acc.	κ	Acc.	κ	Acc.	κ
0.0–0.1	141	63.8	0.27	73.8	0.47	66.0	0.31
0.1–0.2	341	70.1	0.40	75.4	0.51	71.0	0.42
0.2–0.3	350	73.1	0.43	68.0	0.35	72.3	0.41
0.3–0.4	350	72.6	0.45	66.0	0.32	68.6	0.37

Table 7: Effects of semantic similarity (cosSim) on classification (10-fold c.-v.). The scores with discourse features increase as similarity between T_1 and T_2 decreases.

Table 7 shows an analysis of the effect of the semantic features on different training sets in terms of accuracy and kappa scores. These training sets are built according to the cosine similarity between T_1 and T_2 , as follows: for each training instance (pair of nouns), we compute the cosine similarity between the vector representation of T_1 and T_2 ; then, we group instances by intervals and carry out 10-fold c.-v. classification experiments for each subset. The lower the range values, the more dissimilar the translation pairs T_1 and T_2 , and the better the scores of discourse features. Specifically, when the translations are dissimilar, the classifier makes better predictions with the discourse features, i.e. considering a larger context. However, the more similar the words are, the better the local context features, i.e. the surrounding words.

5.3 Extension to Triples of Repeated Nouns

Finally, we consider briefly the case of nouns that appear more than twice. Using our dataset, we identified them as noun pairs that share the same word, i.e. triples of repeated nouns, to which we limit our investigation. There are 129 ZH/EN triples and 138 DE/EN ones.

We defined the following method to determine the translation of such nouns when their baseline translations are different across the two pairs. If T_1 , T_2 and T_3 are the translation candidates, we

aim to find the consistent translation T_c as follows. If two of the T_i are identical, we use this value as T_c , but if they all differ, then we compare the syntactic features of the three source occurrences, and select the one with the highest number of features with highest values, and use its value as T_c . Going back to our classifier, if the decision for a particular instance pair is not ‘none’, then we replace the translations of the instance pairs with T_c .

We tested the method with the three feature types and the four classifiers, i.e. 12 cases per language. On ZH/EN, a small increase of BLEU is observed in 5 cases (0.01), a decrease in two cases (0.02), and no variation in 5 cases. On DE/EN, half of the cases show a small improvement (up to 0.03) and the rest stay the same. The method appears to work better on DE/EN, possibly because the initial accuracy on pairs is lower, but improvements are overall very small. The main conclusion from experimenting with triples, and considering also longer lexical chains of consistent nouns, is that the pairwise method should be replaced by a different type of consistency predictor, which remains to be found.

6 Conclusion and Perspectives

We presented a method for flexibly enforcing consistent translations of repeated nouns, by using a machine learning approach with syntactic and semantic features to decide when it should be enforced. We experimented with Chinese-English and German-English data. To build our datasets, we detected source-side nouns which appeared twice within a fixed distance and were translated differently by MT. Syntactic features were defined based on the complexity of the parse trees containing the nouns, thus capturing which of the two occurrences of a noun is more syntactically bound, while semantic features focused on the similarity between each translated noun and its context. The trained classifiers have shown that they can predict consistent translations above chance, and that, when combined to MT, bridge 50–60% of the gap between the baseline and an oracle classifier.

In future work, we will consider whether neural MT is prone to similar consistency problems, and whether they can be addressed by a similar method. The answer is likely positive, because both PBSMT and NMT assume that consistency simply results from correct individual translations, whereas human translators often take consistency

into account for lexical choice. Moreover, a better consideration of legitimate lexical variation, e.g. using multiple references or human evaluators, should improve the assessment of consistency enforcement strategies.

Acknowledgments

We are grateful for their support to the Swiss National Science Foundation (SNSF) under the Sinergia MODERN project (grant n. 147653, see www.idiap.ch/project/modern/) and to the European Union under the Horizon 2020 SUMMA project (grant n. 688139, see www.summa-project.eu). We thank the three anonymous reviewers for their helpful suggestions.

References

- Leo Breiman. 2001. Random Forests. *Machine Learning*, 45(1):5–32.
- Marine Carpuat and Michel Simard. 2012. The trouble with SMT consistency. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT ’12, pages 442–449.
- Marine Carpuat. 2009. One translation per discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 19–27.
- Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT³: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.
- Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20:37–46.
- Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.
- William A Gale, Kenneth W Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, pages 233–237.
- Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 909–919, Edinburgh.
- Liane Guillou, Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, Mauro Cettolo, Bonnie Webber, and Andrei Popescu-Belis. 2016. Findings of the 2016 WMT shared task on cross-lingual pronoun prediction. In

- Proceedings of the First Conference on Machine Translation (WMT16)*, Berlin, Germany.
- Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-Wide Decoding for Phrase-Based Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL)*, Jeju, Korea.
- Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, and Mauro Cettolo. 2015. Pronoun-focused MT and cross-lingual pronoun prediction: Findings of the 2015 DiscoMT shared task on pronoun translation. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 1–16, Lisbon, Portugal.
- Shuan Fan Huang. 1995. Chinese as a metonymic language. In *In Honor of William Wang: Interdisciplinary studies on Language and Language Change*, pages 223–252, Taipei, Taiwan. Pyramid Press.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbs. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics (ACL), Demonstration Session*, pages 177–180, Prague, Czech Republic.
- Christopher Manning and Dan Klein. 2003. Optimization, MaxEnt Models, and Conditional Estimation without Magic. In *Tutorial at HLT-NAACL and 41st ACL conferences*, Edmonton, Canada and Sapporo, Japan.
- Laura Mascarell, Mark Fishel, Natalia Korchagina, and Martin Volk. 2014. Enforcing consistent translation of German compound coreferences. In *Proceedings of the 12th Konvens Conference*, Hildesheim, Germany.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations*, Scottsdale, Arizona, USA.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA.
- Xiao Pu, Laura Mascarell, Andrei Popescu-Belis, Mark Fishel, Ngoc Quang Luong, and Martin Volk. 2015. Leveraging compounds to improve noun phrase translation from Chinese and German. In *Proceedings of the ACL-IJCNLP 2015 Student Research Workshop*, pages 8–15, Beijing, China.
- J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.
- Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal, September. Association for Computational Linguistics.
- Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15, Uppsala, Sweden.
- Ferhan Ture, Douglas W. Oard, and Philip Resnik. 2012. Encouraging consistent translation choices. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 417–426, Montréal, Canada.

Psycholinguistic Models of Sentence Processing Improve Sentence Readability Ranking

David M. Howcroft¹ and Vera Demberg^{1,2}

¹ Department of Language Science and Technology

² Department of Computer Science

Saarland Informatics Campus, Saarland University

Saarbrücken, Germany

{howcroft, vera}@coli.uni-saarland.de

Abstract

While previous research on readability has typically focused on document-level measures, recent work in areas such as natural language generation has pointed out the need of sentence-level readability measures. Much of psycholinguistics has focused for many years on processing measures that provide difficulty estimates on a word-by-word basis. However, these psycholinguistic measures have not yet been tested on sentence readability ranking tasks. In this paper, we use four psycholinguistic measures: idea density, surprisal, integration cost, and embedding depth to test whether these features are predictive of readability levels. We find that psycholinguistic features significantly improve performance by up to 3 percentage points over a standard document-level readability metric baseline.

1 Introduction

Previous work on readability has classified or ranked texts based on document-level measures such as word length, sentence length, number of different phrasal categories & parse tree depth (Petersen, 2007), and discourse coherence (Graesser et al., 2004), inter alia. However, not all applications that need readability ratings deal with long documents. For many applications in text simplification, computer-aided language learning (CALL) systems, authorship tools, translation, and information retrieval, sentence-level readability metrics are direly needed.

For instance, an automatic text simplification system must begin by asking which portions of a text need to be simplified. To this end, a measure that can assign ratings on a sentence-by-sentence

level can help target simplification only to those sentences which need it most, and such measures also serve to confirm that the resulting ‘simplified’ sentence is in fact simpler than the original sentence.

Similarly, CALL and other pedagogical systems will benefit if it is possible to predict which portions of a text will be harder for students. Authorship tools can offer more specific editorial advice when they know why individual sentences can cause difficulties for readers. Translation tools can aim to preserve not just meaning but also the approximate difficulty of the sentences they are translating or use a sentence-level difficulty metric to target output that is easier to understand. Furthermore, information retrieval systems also benefit when they can return not merely relevant texts, but also texts appropriate to the reading level of the user. Recently there has been an increased interest in sentential models of text difficulty in the automatic text simplification and summarization communities in particular (Vajjala and Meurers, 2014; Macdonald and Siddharthan, 2016).

One area that has produced a lot of research on sentence level processing difficulty is psycholinguistics. Over the past three decades, a number of theories of *human* sentence processing (i.e. reading) have been proposed and validated in a large variety of experimental studies. The most important sentence processing theories have furthermore been implemented based on broad-coverage tools, so that estimates for arbitrary sentences can be generated automatically. For example, eye-tracking studies of reading times on a large corpus of newspaper text have found that measures such as *integration cost* and *surprisal* provide partial explanations for subjects’ reading behavior (Demberg and Keller, 2008).

This paper leverages these implemented measures based on psycholinguistic theories of sen-

tence processing in order to test whether they can help to more accurately score individual sentences with respect to their difficulty. In the process, we evaluate the contributions of the individual features to our models, testing their utility in examining fine-grained distinctions in sentence difficulty. Section 2 reviews the literature on readability in general before we shift to psycholinguistic theories of sentence processing in Section 4. In Section 5 we discuss our methods, including the corpora used, how features were extracted, and the set up for our averaged perceptron models. Section 6 presents our findings which we connect to related work on sentence-level readability models in 3. Finally we offer our conclusions and suggestions for future work in Section 7.

2 Readability

Chall's (1958) comprehensive review of readability research in the first half of the 20th century divides the early work in readability into "survey and experimental studies" and "quantitative associational studies". Studies of the former category took place during the 1930s and 1940s and included surveys of expert and reader opinion as well as experimental studies which manipulated texts according to one variable at a time in order to determine the effects of those variables on readers. The results of these studies suggest that, once you have managed to control for reader interest in the content of a text, the most important factor with respect to its readability is its 'style', e.g. its "scope of vocabulary and...kinds of sentences" (Gray and Leary, 1935, as quoted in (Chall, 1958)).

Our study belongs to the second class, relating the features of a text to its ordering relative to some other texts. The earliest work in this direction was by L. A. Sherman, who proposed a quantitative analysis of text difficulty based on the number of clauses per sentence, among other features (Sherman, 1893). Where Sherman's pedagogical focus was on literature, Lively & Pressey (1923) focused on vocabulary as a bottleneck in science education. Work in this vein led to the development of a number of readability formulae in the mid-20th century¹, including the familiar Flesch-Kincaid Grade-Level score (Kincaid et al., 1975).

¹For a comprehensive review of the literature up to 1958, we recommend (Chall, 1958). For a more recent review of the literature, we recommend Chapter 2 of (Vajjala, 2015). For an introduction to some of the major studies from the 20th century, we recommend the self-published (Dubay, 2007).

These formulae typically use a linear combination of average word length and average sentence length, though some also incorporate a vocabulary-diversity term. The simple, two-feature versions of these models are still widely used, and inspired our BASELINE model.

More recently, Petersen (2007) sought to apply familiar natural language processing techniques to the problem of identifying text difficulty for non-native readers. In particular, she used a number of parse-based features which captured, for example, the average number of noun and verb phrases per sentence and the height of the parse tree. Petersen trained SVM classifiers to classify texts as belonging to one of four primary school grade levels based on the Weekly Reader educational newspaper². These document-level models achieved F -scores in the range of 0.5 to 0.7, compared to the F -scores between 0.25 and 0.45 achieved by the Flesch-Kincaid Reading Ease score for the same texts.

Recent work has also looked at features related to discourse and working memory constraints. Feng et al. (2009) worked on a model of readability for adults with intellectual disabilities. Considering working memory constraints, they extracted features related to the number of entities mentioned in a document and the 'lexical chains' (Galley and McKeown, 2003) that connected them. They found that their features resulted in a better correlation (Pearson's $r = -0.352$) compared to both Flesch-Kincaid score ($r = -0.270$) and a number of 'basic' linguistic features based on those used by Petersen & Ostendorf (2009) ($r = -0.283$).³

Coh-Metrix (Graesser et al., 2004) also includes a number of measures related to discourse coherence, for example. Such features are not suited to the problem of determining the difficulty of sentences in isolation, but they have also been shown to better predict readability for second-language learners compared to 'traditional' readability measures like those described above (Crossley et al., 2011).

²<http://www.weeklyreader.com>

³Correlations here are negative because Feng et al. correlated predicted reading levels with the performance of adults with intellectual disabilities on comprehension tests. The adults with disabilities are expected to perform worse on the comprehension test as the grade level of the text increases.

3 Measuring Sentence Complexity

Classification Few studies to date have addressed sentence-level readability for English. Napoles & Dredze (2010) built their own corpus with documents from English and Simple English Wikipedia to train both document- and sentence-level classifiers. Using bag-of-words features, unigram and bigram part-of-speech features, type-token ratio, the proportion of words appearing on a list of easier words, and parse features similar to Petersen's, their binary classifier achieved an accuracy of 80.8% on this task. The structure of this task, however, is not suited to text simplification applications, because the sentences are not controlled for meaning. Classifying a sentence in isolation as more likely to be from Simple Wikipedia or English Wikipedia is not as useful as a model trained to differentiate sentences carrying the same meaning. This work is not directly comparable to that of Vajjala & Meurers (2012; 2014) or subsequent work on ranking sentences by their complexity due to the differences in choice of corpora and task structure.

In the medical domain, Kauchak et al. (2014) also looked at sentence-level classification, identifying sentences as being either simple or difficult. Their features included word length, sentences length, part-of-speech counts, average unigram frequencies and standard deviation, and the proportion of words not on a list of the five thousand most frequent words as well as three domain-specific features based on an ontology of medical terminology.

Ranking Vajjala & Meurers (Vajjala and Meurers, 2014; Vajjala, 2015) were the first to look at *ranking* sentences rather than classifying them, having observed that the distributions of predicted reading levels across the two subcorpora of the Parallel Wikipedia corpus (Zhu et al., 2010, PWKP) were different. While the Simple English portion of the corpus was clearly skewed toward the lower grade levels, it appears that the English portion of the corpus was evenly distributed across all grade levels, making binary-classification difficult.

This led Vajjala & Meurers to develop a ranking model using the predicted reading levels from a multiclass classifier trained on whole documents. For each sentence pair, they assumed that the English Wikipedia sentence should be classified at a higher level than the Simple English

Wikipedia sentence. Using a hard cut-off (i.e. $rank(sent_{english}) > rank(sent_{simple})$), their model achieved about 59% accuracy, although this improved to 70% by relaxing the inequality constraint to include equality. Based on the finding that 30% of sentence pairs from the PWKP corpus are incorrectly ranked despite lying within one reading level of each other, we hypothesize that finer-grained distinctions may be necessary to tease apart the differences in related pairs of sentences.

Offline Psycholinguistic Features While Vajjala & Meurers (2012; 2014) do use some psycholinguistically-motivated features, their features are primarily lexical in nature and therefore complementary to ours, which depend on the sentence processing context. They drew psycholinguistic features from the MRC psycholinguistic database (Wilson, 1988), including word familiarity, concreteness, imageability, meaningfulness, and age of acquisition. These features were coupled with a second age of acquisition database and values related to the average number of senses per word.

Towards online considerations More recently, Ambati et al. (2016) used an incremental parser to extend Vajjala & Meurers work. Since human processing is incremental, they reasoned, features from an incremental parser might be more informative than features extracted from a non-incremental parser. To this end, they used the incremental derivations from a combinatorial grammar (CCG) parser. Ambati et al. ran several models on the English and Simple English Wikipedia data set (Hwang et al., 2015, ESEW): one using only the syntactic features from (Vajjala and Meurers, 2014); another (INCCCG) using only features from the incremental parser; and INCCCG+, incorporating morpho-syntactic and psycholinguistic features from (Vajjala and Meurers, 2014). At the sentence level, they include sentence length, number of CCG constituents in the final parse, and the depth of the CCG derivation. They also use count features for the number of times each CCG derivation rule is applied (e.g. forward application, type-raising). Finally, they include counts of different CCG syntactic categories as well as the average 'complexity' of the syntactic categories. While the parser they use is inspired by human behavior, in that it is an incremental parser, these features do not re-

late to any specific linguistic theory of sentence processing.

The work presented here is most comparable to that of Vajjala & Meurers and Ambati et al., as we all address the problem of ranking sentences according to their linguistic complexity. Our study is the only one of the three to examine features based on theories of online sentence processing. Ambati et al. (2016) provide accuracy information for their own features as well as Vajjala & Meurers' (2014) features on the English and Simple English Wikipedia corpus (ESEW) which we use, but used a 60-20-20 training-dev-test split where we used 10-fold cross-validation, making the results not directly comparable.

4 Theories of Online Sentence Processing

For our purposes, we focus on readability as reading ease and on linguistic constraints in particular, rather than constraints of medium (relating to e.g. legibility), reader interest, or comprehensibility. Without directly modeling comprehensibility, we assume that making material easier to read will also make it easier to comprehend. Here we focus on four psycholinguistic theories of human sentence processing: idea density, surprisal, integration cost, and embedding depth.

Kintsch (1972) defined **propositional idea density** as the ratio of propositions or ideas to words in the sentences.⁴ Keenan & Kintsch conducted two different experiments in order to examine free reading behavior as well as subjects' performance in speeded reading conditions. They found that "the number of propositions [in a text] had a large effect upon reading times, [but] it could only account for 21% of their variance" when subjects were allowed to read freely. Subjects' overall recall was worse for more dense texts in the speeded reading condition. In addition to effects of idea density, they found that propositions which were presented as surface-form modifiers (as opposed to, e.g., main verbs) were "very poorly recalled" and that propositions playing a subordinate role relative to another proposition were also less-well recalled. Finally, propositions involving a proper name were generally recalled better than similar propositions involving, e.g., a common noun.

While Kintsch & Keenan (1973) looked at the

⁴ This notion of idea density is closely related to Perfetti's (1969) notion of *lexical density* insofar as both are related to the number of so-called *content words* in the text.

influence of propositional idea density on reading times and recall for both individual sentences as well as short paragraphs, work since the 1970s has been limited to the level of multiple sentences and used primarily as an indicator of cognitive deficits (Ferguson et al., 2014; Bryant et al., 2013; Farias et al., 2012; Riley et al., 2005). This paper returns to the examination of idea density's applicability for individual sentences.

Surprisal, on the other hand, has been widely examined in theories of language comprehension at a variety of levels, including the word- and sentence-levels. **Surprisal** is another word for Shannon (1948) information, operationalized in linguistics as the probability of the current word conditioned on the preceding sequence of words:

$$\text{surprisal}(w_n) = -\log(P(w_n|w_1 \dots w_{n-1})) \quad (1)$$

where w_i is the i^{th} word in the sentence and $P(w_1 \dots w_i)$ denotes the probability of the sequence of i words $w_1 \dots w_i$.

One reason psycholinguists consider surprisal as a factor in sentence processing difficulty is that it makes sense in a model of language users as rational learners. Levy (2008) argues the rational reader's attention must be spread across all possible analyses for the sentence being observed. Based on prior experience, the reader expects some analyses to be more probable than others and therefore allocates more resources to those analyses. In this analysis, surprisal is derived as a measure of the cost paid when the reader misallocates resources: when a new word invalidates a highly probable analysis, the reader has effectively 'wasted' whatever resources were allocated to that analysis. The notion of surprisal is also used in theories of language production, see the Uniform Information Density hypothesis (Jaeger, 2006; Levy and Jaeger, 2007; Jaeger, 2010, UID).

While surprisal focuses on predictability effects in sentence processing, Gibson's (1998; 2000) **Dependency Locality Theory** (DLT) focuses on the memory cost of recalling referents and integrating new ones into a mental representation. DLT proposes that the the distance between syntactic heads and dependents, measured by the number of intervening discourse referents, approximates the difficulty that the listener or reader will have integrating the two units. This model maintains that the act of creating a new discourse referent and holding it in memory makes it more difficult to recall

a previous discourse referent and connect that discourse referent to the current one.⁵

In addition to *integration cost*, DLT proposes a *storage cost* associated with the number of open dependencies that must be maintained in memory. The notion of connected components in van Schijndel et al.'s (2012; 2013) incremental parsing model picks up this idea. Related models were also suggested earlier by Yngve (1960) and Miller's (1956a; 1956b) whose work was based on results showing that human working memory is limited to 7 ± 2 items. Yngve's mechanistic, incremental model of language production considered the evaluation of phrase structure grammars (PSGs) in a system with finite memory, exploring the structure speakers must keep track of during production and how grammars might be structured to avoid overtaxing working memory.

Van Schijndel et al. develop this idea further in the context of a hierarchical sequence model of parsing. In this incremental model of parsing, at each stage the reader has an *active* state (e.g. S for sentence) and an *awaited* state (e.g. VP for verb phrase).⁶ At each new word, the parser must decide between continuing to analyze the current connected component or hypothesizing the start of a new one.⁷

These measures provide an idealized representation of the number of different states a human parser must keep track of at any point in time. We refer to this number of states as the **embedding depth** of a sentence at a particular word, and the ModelBlocks parser of van Schijndel et al. (2012) calculates this number of states averaged over the beam of currently plausible parses. Also of interest is the *embedding difference*, which is the embedding depth at the present word relative to the previous word, elaborated upon in the following example.

Consider the state described above (i.e. that of being in the active state S and awaiting state VP) might be reached after a reader has observed a noun phrase, resulting in the state S/VP. This

means that the word sequence observed so far will be consistent with a sentence if the reader now observes a verb phrase. If, however, the next word in the input is inconsistent with the start of a verb phrase (e.g. the relative clause marker *that*), then this parse will be ruled out and another must be considered. At this point the parser must hypothesize the beginning of a new connected component, i.e. a new syntactic substructure that must be completed before continuing to parse the top-level of the sentence. Therefore, the parser must now keep track of two states: (1) the fact that we are still looking for a VP to complete the overall sentence; and (2) the fact that we now have a relative clause to parse before we can complete the current NP. In this example, we are at embedding depth 1 or 0 up until we encounter the word *that*, which increases the embedding depth by 1, resulting in a nonzero embedding difference score.

4.1 Experimental Evidence

We have already explained the experimental findings of Kintsch & Keenan (1973) with respect to idea density, but what behavioral evidence is there to suggest that the remaining theories are valid?

Demberg & Keller (2008) examined the relationship between both surprisal and integration cost and eye-tracking times in the Dundee corpus (Kennedy and Pynte, 2005). Demberg & Keller found that increased surprisal significantly correlated with reading times. Although they found that integration cost did not significantly contribute to predicting eye-tracking reading times in general, its contribution was significant when restricted to nouns and verbs. They also found that surprisal and integration cost were uncorrelated, suggesting that they should be considered complementary factors in a model of reading times. Another eye-tracking study divided surprisal into lexical and syntactic components, finding that lexical surprisal was a significant factor but not syntactic surprisal (Roark et al., 2009).

Wu et al. (2010) examined surprisal, entropy reduction, and embedding depth in a study of psycholinguistic complexity metrics. Their study of the reading times of 23 native English speakers reading four narratives indicated that embedding difference was a significant predictor of reading times for closed class words. Moreover, this contribution was independent of the contribution of surprisal, indicating that the two measures are

⁵ Gildea & Temperley (2010) measure dependencies in terms of word span, such that adjacent words have a dependency length of one. This approach produces similar difficulty estimates nouns and verbs, with the caveat that distances are systematically increased, and is defined for all words in a sentence.

⁶ In Combinatory Categorical Grammar notation, this state is denoted S/VP.

⁷ These connected components are the parsing analogues to the constituents awaiting expansion in Yngve's analysis.

capturing different components of the variance in reading times. Since integration cost was a significant predictor of reading times for nouns and verbs (i.e. not closed class words) and embedding depth was a significant predictor of reading times for closed class words, integration cost and embedding depth should also be complementary to each other.

5 Methods

5.1 Corpora

We used two corpora in this work. The **English and Simple English Wikipedia** corpus of Hwang et al. (2015, ESEW) is a new corpus of more than 150k sentence pairs designed to address the flaws of the Parallel Wikipedia Corpus of Zhu et al. (2010, PWKP), which was previously dominant in work on text simplification, by using a more sophisticated method of aligning pairs of English and Simple English sentences. We used the section labeled as having ‘good’ alignments for our work and assumed that, in every sentence pair, the Simple English sentence should be ranked as easier than the English sentence ($rank=1 < rank=2$ in Table 1). This provides a large corpus with noisy labels, as there are likely to be instances where the English and Simple English sentences are not substantially different or the English sentence is the easier one.⁸

For a more controlled corpus, we use Vajjala’s (2015) **One Stop English** (OSE) corpus. This corpus consists of 1577 sentence triples, drawn from news stories edited to three difficulty levels: elementary, intermediate, and advanced. Vajjala used $TF * IDF$ and cosine similarity scores to align sentences from stories drawn from `onestopenglish.com`. While One Stop English does not publish an explanation of their methods for creating these texts, they are at least created by human editors for pedagogical purposes, so the labels should be more consistent and reliable than those associated with the ESEW corpus.

The three levels of *OSE* make it possible to compare system performance on sentence pairs which are close to one another in difficulty (e.g. ‘advanced’ versus ‘intermediate’ sentences) with

⁸Indeed, 37,095 of the 154,805 sentence pairs have the same sentence for both English and Simple English Wikipedia and were therefore excluded from our experiments.

performance on pairs which are further apart, as with ‘advanced’ sentences paired with their ‘elementary’ counterparts. In this paper we will refer to the pairs of advanced and elementary sentences as OSE_{far} , the remaining pairs as OSE_{near} , and the full OSE dataset as OSE_{all} . An example triple of sentences from the corpus is given in Table 2.

5.2 Feature Extraction and Feature Sets

We used two parsers to extract 22 features from the corpora. The `ModelBlocks` parser provided features based on surprisal and embedding depth while the Stanford parser⁹ provided the dependency parses used to calculate integration cost and idea density features. Both parsers are trained and perform near the state of the art on the standard sections of the Wall Street Journal section of the Penn Treebank.

From `ModelBlocks`’ complexity feature extraction mode, we took the lexical and syntactic surprisal features. We used the average lexical surprisal and average syntactic surprisal as idealized measures of the channel capacity required to read a sentence. While this underestimates the channel capacity required to process a sentence, it is at least internally consistent, insofar as a sentence with higher average surprisal overall is likely to require a higher channel capacity as well. We also used the maximum of each form of surprisal as a measure of the maximum demand on cognitive resources. These features comprise the `SURPRISAL` model.

We also calculated average and maximum values for the embedding depth and embedding difference output from `ModelBlocks`. The average provides an estimate of the typical memory load throughout a sentence, while the (absolute) embedding difference is a measure of how many times a reader needs to push or pop a connected component to or from their memory store. These features comprise the `EMBEDDING` model.

To extract the remaining features, we first ran the Stanford dependency parser on both corpora. The program `icy-parses` uses part-of-speech tags and head-dependent relations to determine the total, average, and maximum integration cost across a sentence. Here average integration cost functions as another kind of memory load estimate while the maximum value models the most-

⁹<http://nlp.stanford.edu/software/lex-parser.shtml>

Rank	Sentence
2	Gingerbread was brought to Europe in 992 by the Armenian monk Gregory of Nicopolis -LRB- Gregory Makar -RRB- -LRB- Grégoire de Nicopolis -RRB- .
1	Armenian monk Gregory of Nicopolis -LRB- Gregory Makar -RRB- -LRB- Grgoire de Nicopolis -RRB- brought ginger bread to Europe in 992 .

Table 1: Example sentences from English (2) and Simple (1) English Wikipedia.

Rank	Sentence
3	It is a work-hard, play-hard ethic that many of the world’s billionaires might subscribe to but it would be a huge change for most workers and their employers.
2	It is a ‘work-hard, play-hard’ way of thinking that many of the world’s billionaires might agree with but it would be a huge change for most workers and their employers.
1	Many of the world’s billionaires might agree with this way of thinking but it would be a very big change for most workers and their employers.

Table 2: Example sentences from One Stop English, at levels advanced (3), intermediate (2), and elementary (1). The pair 3–1 is in OSE_{far} , the pairs 3–2 and 2–1 are in OSE_{near} , and all three pairs are in OSE_{all} .

difficult-to-integrate point in the sentence. These features comprise the INTEGRATIONCOST model.

Finally, we use a modified version of the `IDD3` library from Andre Cunha (Cunha et al., 2015) to extract idea density decomposed across three types of propositional idea: predications, modifications, and connections.¹⁰ Here we use only averaged features, as the crucial measure is the idea *density* rather than the raw number of ideas being expressed. These features comprise the IDEADENSITY model.

As a point of comparison for these models, we created a `BASELINE` which used only sentence length and the average word length as features.

We also created models based on features grouped by the parser used to extract them: `SURPRISAL+EMBED` for the `ModelBlocks` parser and `IDEA+INTEGRATION` for the Stanford parser. While `ModelBlocks` achieves competitive accuracies, it is much slower than other state-of-the-art parsers available today. Therefore we wanted to provide a point of comparison regarding the relative utility of these parsers: grouping features by parser allows us to assess the trade-off between model accuracy and the time necessary for feature extraction.

Finally, we considered combinations of the parser-grouped features with the baseline (`BASE+SURPRISAL+EMBED` and `BASE+IDEA+INTEGRATION`) and a

¹⁰Code available at: <https://github.com/dmhowcroft/idd3>.

`FULLMODEL` using the baseline features and all of the psycholinguistic features.

Replication The scripts required for replication are available at <https://github.com/dmhowcroft/eacl2017-replication>. This includes pointers to the corpora, pre-processing scripts and settings for the parsers, as well as scripts for feature extraction and running the averaged perceptron model.

5.3 Ranking as Classification

In order to rank sentences, we need some way of generating a complexity score for each sentence. Using a perceptron model allows us to train a simple linear scoring model by converting the ranking task into a classification task.

Suppose we have two sentences s_1 and s_2 with feature vectors \mathbf{s}_1 and \mathbf{s}_2 such that s_1 is more complex than s_2 . Then we want to train a perceptron model such that

$$\text{score}(s_1) > \text{score}(s_2) \quad (2)$$

$$\mathbf{W} \cdot \mathbf{s}_1 > \mathbf{W} \cdot \mathbf{s}_2 \quad (3)$$

$$\mathbf{W} \cdot (\mathbf{s}_1 - \mathbf{s}_2) > 0 \quad (4)$$

We refer to the vector $\mathbf{s}_1 - \mathbf{s}_2$ as a vector of *difference features*. In order to train the model, we take all pairs of sentences present in a given corpus and create a difference vector as above. In half of the cases, we flip the sign of the difference vector, creating a binary classification task with balanced classes. The learning problem is now to



Figure 1: Results on the ESEW corpus for each set of psycholinguistic features individually (first 4 columns) and altogether (5th column), with the feature sets based on the `ModelBlocks` and Stanford parsers in the last two columns.

classify each difference vector based on whether the first term in the difference was the ‘easier’ or the ‘harder’ sentence

Note that the benefit to this approach is that the resulting weight vector \mathbf{W} learned via the classification task can be used directly to score individual sentences as well, with the expectation that higher scores will correspond to more difficult sentences.

We use an averaged perceptron model (Collins, 2002) implemented in Python as our classifier.

6 Analysis & Results

The feature sets for individual psycholinguistic theories only achieve accuracies between 55% and 65% (see the first 4 columns of Fig. 1). Combining all of these features into the `PSYCHOLINGUISTIC` model improves performance to nearly 70% (column 5). Looking at the feature sets grouped by parser (columns 6 and 7), we see that the combination of surprisal and embedding depth (from the `ModelBlocks` parser) significantly outperforms the combination of integration cost and idea density (from the Stanford Parser). However, the strength of the features derived from `ModelBlocks` seems to be primarily driven by the `EMBEDDING` features, while the strength of the dependency-parse-derived features appears to stem from `INTEGRATIONCOST`.

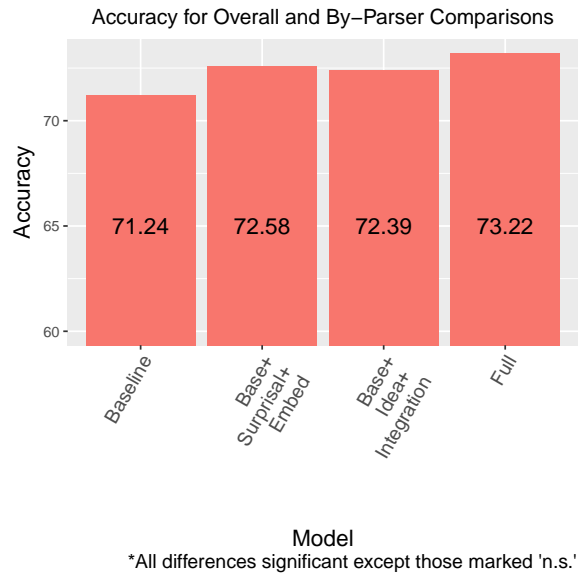
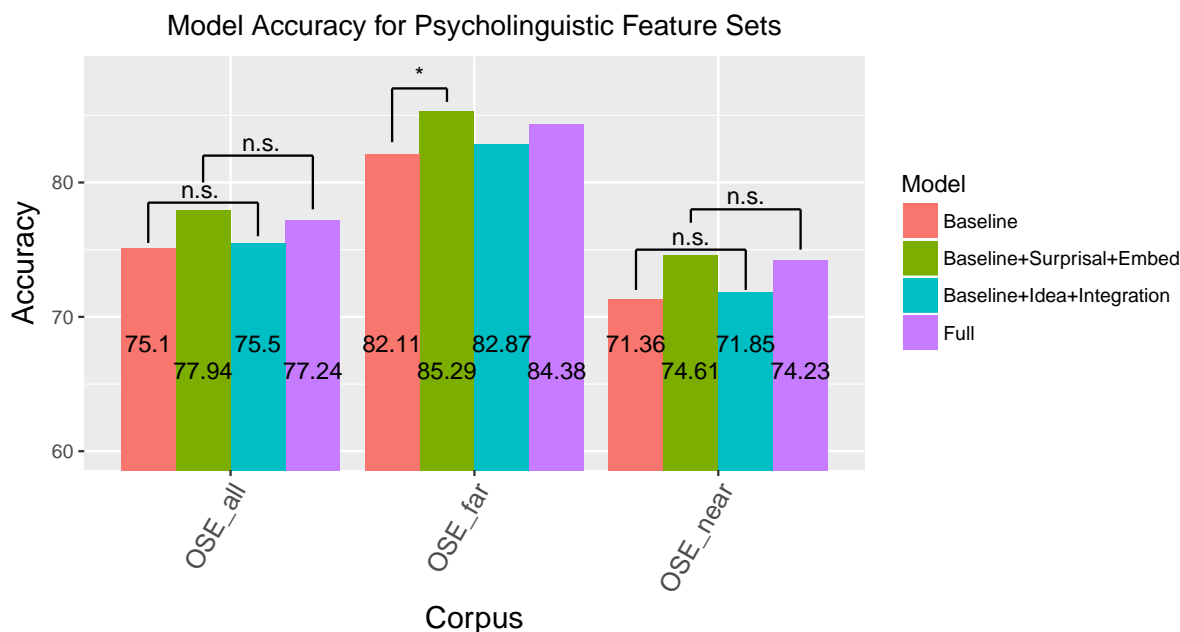


Figure 2: Results for the baseline model, our two parser-grouped feature sets, and the full model on the ESEW corpus.

Moving to Figure 2, we see that our `BASELINE` features achieved an accuracy of 71.24%, despite using only average word length and sentence length. This is 1.48 percentage points higher than the 69.76% accuracy of the `PSYCHOLINGUISTIC` model, which includes surprisal, embedding depth, integration cost, and idea density. However, the `FULL` model (column 4) outperforms the `BASELINE` by a statistically significant¹¹ 1.98 percentage points ($p \ll 0.01$). This confirms our primary hypothesis: psycholinguistic features based on online sentence processing can improve models of sentence complexity beyond a simple baseline.

To address the secondary hypothesis, we turn to the `OSE` data in Figure 3. The best model for this corpus uses the baseline features combined with embedding depth and surprisal features extracted from `ModelBlocks`. In both OSE_{far} and OSE_{near} we gain about 3 points over the baseline when adding these features (3.18 and 3.25 points, respectively), which is similar to the gains for the `FULL` model over the baseline. The fact that the increase in performance between the `BASELINE` model and the best performing model does not differ between the OSE_{near} and the OSE_{far} datasets suggests a lack of support for our secondary hypothesis that these features are espe-

¹¹ Using McNemar’s (1947) test throughout, as is standard for paired samples like ours, with Bonferroni correction where appropriate.



*All differences significant except those marked 'n.s.'

Figure 3: Results for the baseline model, our two parser-grouped feature sets, and the full model on the OSE corpus, with additional breakdown by level proximity.

cially helpful for distinguishing items of similar difficulty levels.

These results warrant a full comparison to the work of Ambati et al. (2016), despite the differences in our evaluation sets. Ambati et al. found that their features based on incremental CCG derivations achieved an accuracy of 72.12%, while the offline psycholinguistic features of Vajjala & Meurers came in at 74.58%, 1.36 percentage points better than our 73.22%. Finally, a model combining all of Vajjala & Meurers features with the incremental CCG features achieved a performance of 78.87%. Since the features examined in our study are complementary to those proposed by these two previous studies, a model combining all of these features should further improve in accuracy.

7 Conclusion

We examined features for the ranking of sentences by their complexity, training linear models on two corpora using features derived from psycholinguistic theories of online sentence processing: idea density, surprisal, integration cost, and embedding depth.

Surprisal coupled with embedding depth and our baseline features (average word length & sentence length) performed as well as the full model

across all subsets of the OSE corpus. Integration cost and idea density were less effective, suggesting that the gain in speed from running a faster dependency parser may not be worth it. Instead, it is necessary to use the slower `ModelBlocks` parser to extract the more useful features.

Overall, our strongest model combined the baseline features and the online psycholinguistic features. Because these features are complementary to features which have been explored in other work (Vajjala and Meurers, 2014; Ambati et al., 2016), the next step in future work is to combine all of these features and conduct a more comparison between the features proposed here and those examined in earlier work. In the meantime, we have demonstrated that features derived from psycholinguistic theories of sentence processing can be used to improve models for ranking sentences by readability.

Acknowledgments

Thanks are due to Matthew Crocker, Michael White, Eric Fosler-Lussier, William Schuler, Detmar Meurers, Marten van Schijndel, and Sowmya Vajjala for discussions and guidance during the development of this work. We are supported by DFG collaborative research center SFB 1102 ‘Information Density and Linguistic Encoding’.

References

- Ram Bharat Ambati, Siva Reddy, and Mark Steedman. 2016. Assessing relative sentence complexity using an incremental ccg parser. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1051–1057. Association for Computational Linguistics.
- Lucy Bryant, Elizabeth Spencer, Alison Ferguson, Hugh Craig, Kim Colyvas, and Linda Worrall. 2013. Propositional Idea Density in aphasic discourse. *Aphasiology*, (July):1–18, jun.
- Jeanne S. Chall. 1958. *Readability: an appraisal of research and application*. The Ohio State University, Columbus, OH, USA.
- Michael Collins. 2002. Ranking Algorithms for NamedEntity Extraction: Boosting and the Voted Perceptron. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 489–496, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Scott A. Crossley, David B. Allen, and Danielle S. McNamara. 2011. Text readability and intuitive simplification: A comparison of readability formulas. *Reading in a Foreign Language*, 23(1):84–101.
- Andre Luiz Verucci Da Cunha, Lucilene Bender De Sousa, Leticia Lessa Mansur, and Sandra Maria Aluísio. 2015. Automatic Proposition Extraction from Dependency Trees: Helping Early Prediction of Alzheimer’s Disease from Narratives. *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, pages 127–130.
- Vera Demberg and Frank Keller. 2008. Data from Eye-tracking Corpora as Evidence for Theories of Syntactic Processing Complexity. *Cognition*, 109(2):193–210.
- William H. Dubay. 2007. *Unlocking Language: The Classic Readability Studies*.
- Sarah Tomaszewski Farias, Vineeta Chand, Lisa Bonnici, Kathleen Baynes, Danielle Harvey, Dan Mungas, Christa Simon, and Bruce Reed. 2012. Idea density measured in late life predicts subsequent cognitive trajectories: Implications for the measurement of cognitive reserve. *Journals of Gerontology - Series B Psychological Sciences and Social Sciences*, 67 B(6):677–686.
- Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. 2009. Cognitively motivated features for readability assessment. In *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 229–237.
- Alison Ferguson, Elizabeth Spencer, Hugh Craig, and Kim Colyvas. 2014. Propositional Idea Density in women’s written language over the lifespan: Computerized analysis. *Cortex*, 55(1):107–121, jun.
- M. Galley and K. McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proc. of the 18th International Joint Conference on Artificial Intelligence*, pages 1486–1488.
- Edward Gibson. 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68(1):1–76.
- Edward Gibson. 2000. The Dependency Locality Theory: A Distance-Based Theory of Linguistic Complexity. In Y Miyashita, A Marantz, and W O’Neil, editors, *Image, Language, Brain*, chapter 5, pages 95–126. MIT Press, Cambridge, Massachusetts.
- Daniel Gildea and David Temperley. 2010. Do Grammars Minimize Dependency Length? *Cognitive Science*, 34:286–310.
- Arthur C. Graesser, Danielle S. McNamara, Max M. Louwerse, and Zhiqiang Cai. 2004. Coh-matrix: analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers*, 36(2):193–202.
- William S. Gray and Bernice E. Leary. 1935. *What makes a book readable*. University of Chicago Press, Chicago, Illinois, USA.
- William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning Sentences from Standard Wikipedia to Simple Wikipedia. In *Proc. of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, Colorado, USA.
- T. Florian Jaeger. 2006. *Redundancy and Syntactic Reduction in Spontaneous Speech*. Unpublished dissertation, Stanford University.
- T. Florian Jaeger. 2010. Redundancy and reduction: speakers manage syntactic information density. *Cognitive Psychology*, 61(1):23–62, aug.
- David Kauchak, Obay Mouradi, Christopher Pentoney, and Gondy Leroy. 2014. Text simplification tools: Using machine learning to discover features that identify difficult text. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 2616–2625.
- Alan Kennedy and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research*, 45(2):153–168.
- J. Peter Kincaid, Robert P. Fishburne, Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel. Technical report, Naval Technical Training Command, Memphis - Millington, TN, USA.
- Walter Kintsch and Janice Keenan. 1973. Reading Rate and of Propositions Retention as a Function of the Number in the Base Structure of Sentences. *Cognitive Psychology*, 5:257–274.

- Walter Kintsch. 1972. Notes on the structure of semantic memory. In Endel Tulving and Wayne Donaldson, editors, *Organization of memory*, pages 247–308. Academic Press, New York, New York, USA.
- Roger Levy and T. Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. *Advances in Neural Information Processing Systems 20 (NIPS)*.
- Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–77, mar.
- Bertha A. Lively and S. L. Pressey. 1923. A Method for Measuring the ‘Vocabulary Burden’ of Textbooks. *Educational Administration and Supervision*, IX:389–398.
- Iain Macdonald and Advait Siddharthan, 2016. *Proceedings of the 9th International Natural Language Generation conference*, chapter Summarising News Stories for Children, pages 1–10. Association for Computational Linguistics.
- Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.
- George A. Miller. 1956a. Human memory and the storage of information. *IRE Transactions on Information Theory (IT-2)*, 2(3):129–137, Sep.
- George A. Miller. 1956b. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*.
- Courtney Napoles and Mark Dredze. 2010. Learning Simple Wikipedia : A Cogitation in Ascertaining Abecedarian Language. *Computational Linguistics*, (June):42–50.
- Charles A. Perfetti. 1969. Lexical density and phrase structure depth as variables in sentence retention. *Journal of Verbal Learning and Verbal Behavior*, 8(6):719–724.
- Sarah E. Petersen and Mari Ostendorf. 2009. A machine learning approach to reading level assessment. *Computer Speech and Language*, 23:89–106.
- Sarah E. Petersen. 2007. *Natural Language Processing Tools for Reading Level Assessment and Text Simplification for Bilingual Education*. PhD thesis, University of Washington.
- Kathryn P. Riley, David A. Snowdon, Mark F. Desrosiers, and William R. Markesbery. 2005. Early life linguistic ability, late life cognitive function, and neuropathology: findings from the Nun Study. *Neurobiology of Aging*, 26(3):341–347, Mar.
- Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 324–333. Association for Computational Linguistics.
- Claude E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423.
- L. A. Sherman. 1893. *Analytics of Literature*. Ginn & Company, Boston, Massachusetts, USA.
- Sowmya Vajjala and Detmar Meurers. 2012. On Improving the Accuracy of Readability Classification using Insights from Second Language Acquisition. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications (BEA7)*. Association for Computational Linguistics.
- Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. In *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden. Association for Computational Linguistics.
- Sowmya Vajjala. 2015. *Analyzing Text Complexity and Text Simplification: Connecting Linguistics, Processing and Educational Applications*. Phd thesis, Eberhard Karls Universitaet Tuebingen.
- Marten van Schijndel, Andy Exley, and William Schuler, 2012. *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2012)*, chapter Connectionist-Inspired Incremental PCFG Parsing, pages 51–60. Association for Computational Linguistics.
- Marten van Schijndel, Andy Exley, and William Schuler. 2013. A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–40.
- Michael D. Wilson. 1988. The MRC Psycholinguistic Database: Machine Readable Dictionary, Version 2. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–11.
- Stephen Wu, Asaf Bachrach, Carlos Cardenas, and William Schuler. 2010. Complexity metrics in an incremental right-corner parser. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1189–1198. Association for Computational Linguistics.
- Victor H. Yngve. 1960. A Model and an Hypothesis for Language Structure. *American Philosophical Society*, 104(5):444–466.
- Zheming Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361. Coling 2010 Organizing Committee.

Web-Scale Language-Independent Cataloging of Noisy Product Listings for E-Commerce

Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabrizio, and Ankur Datta

Rakuten Institute of Technology, Boston, MA, 02110 - USA

{pradipto.das, ts-yandi.xia, aaron.levine}@rakuten.com

{giuseppe.difabrizio, ankur.datta}@rakuten.com

Abstract

The cataloging of product listings through taxonomy categorization is a fundamental problem for any e-commerce marketplace, with applications ranging from personalized search recommendations to query understanding. However, manual and rule based approaches to categorization are not scalable. In this paper, we compare several classifiers for categorizing listings in both English and Japanese product catalogs. We show empirically that a combination of words from product titles, *navigational breadcrumbs*, and *list prices*, when available, improves results significantly. We outline a novel method using correspondence topic models and a lightweight manual process to reduce noise from mis-labeled data in the training set. We contrast linear models, gradient boosted trees (GBTs) and convolutional neural networks (CNNs), and show that GBTs and CNNs yield the highest gains in error reduction. Finally, we show GBTs applied in a language-agnostic way on a large-scale Japanese e-commerce dataset have improved taxonomy categorization performance over current state-of-the-art based on deep belief network models.

1 Introduction

Web-scale e-commerce catalogs are typically exposed to potential buyers using a taxonomy categorization approach where each product is categorized by a label from the taxonomy tree. Most e-commerce search engines use taxonomy labels to optimize query results and match relevant listings to users' preferences (Ganti et al., 2010). To illustrate the general concept, consider Fig. 1. A merchant pushes new men's clothing listings to

an online catalog infrastructure, which then organizes the listings into a taxonomy tree. When a user searches for a denim brand, "*DSquared2*", the search engine first has to understand that the user is searching for items in the "*Jeans*" category. Then, if the specific items cannot be found in the inventory, other relevant items in the "*Jeans*" category are returned in the search results to encourage the user to browse further. However, achieving good product categorization for e-commerce market-places is challenging.

Commercial product taxonomies are organized in tree structures three to ten levels deep, with thousands of leaf nodes (Sun et al., 2014; Shen et al., 2012b; Pyo et al., 2016; McAuley et al., 2015). Unavoidable human errors creep in while uploading data using such large taxonomies, contributing to mis-labeled listing noise in the data set. Even EBay, where merchants have a unified taxonomy, reported a 15% error rate in categorization (Shen et al., 2012b). Furthermore, most e-commerce companies receive millions of new listings per month from hundreds of merchants composed of wildly different formats, descriptions, prices and meta-data for the *same products*. For instance, the two listings, "*University of Alabama all-cotton non iron dress shirt*" and "*U of Alabama 100% cotton no-iron regular fit shirt*" by two merchants refer to the same product.

E-commerce systems trade-off between classifying a listing directly into one of thousands of leaf node categories (Sun et al., 2014; ?) and splitting the taxonomy at predefined depths (Shen et al., 2011; ?) with smaller subtree models. In the latter case, there is another trade-off between the number of hierarchical subtrees and the propagation of error in the prediction cascade. Similar to (Shen et al., 2012b; Cevahir and Murakami, 2016), we classify product listings in two or three steps, depending on the taxonomy size. First, we predict the top-level category and then clas-

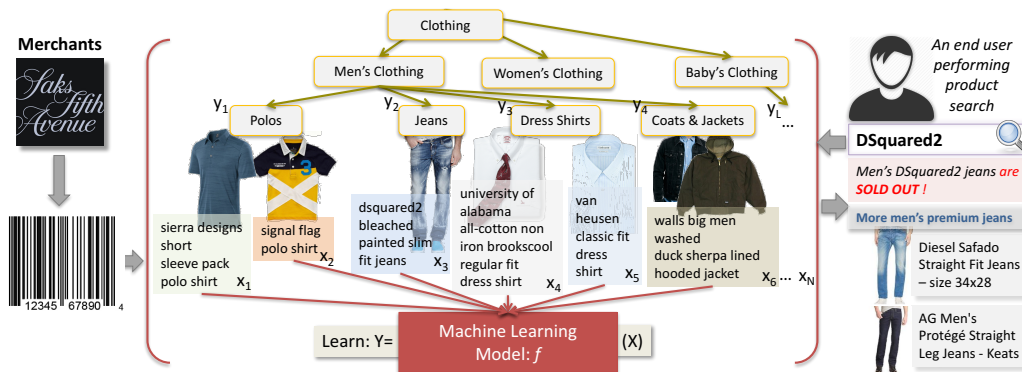


Figure 1: E-commerce platform using taxonomy categorization to understand query intent, match merchant listings to potential buyers as well as to prevent buyers from navigating away on search misses.

sify the listings using another one or two levels of subtree models selected by the previous predictions. For our large-scale taxonomy categorization experiments on product listings, we use two in-house datasets,¹ a publicly available Amazon product dataset (McAuley et al., 2015), and a publicly available Japanese product dataset.²

Our paper makes several contributions: 1) We perform large-scale comparisons with several robust classification methods and show that Gradient Boosted Trees (GBTs) (Friedman, 2000; ?) and Convolutional Neural Networks (CNNs) (LeCun and Bengio, 1995; ?) perform substantially better than state-of-the-art linear models (Section 5). We further provide analysis of their performance with regards to imbalance in our datasets. 2) We demonstrate that using both *listing price* and *navigational breadcrumbs* – the branches that merchants assign to the listings in web pages for navigational purposes – boost categorization performance (Section 5.3). 3) We effectively apply correspondence topic models to detect and remove mis-labeled instances in training data with minimal human intervention (Section 5.4). 4) We empirically demonstrate the effectiveness of GBTs on a large-scale Japanese product dataset over a recently published state-of-the-art method (Cevahir and Murakami, 2016), and in turn the otherwise language-agnostic capabilities of our system given a language-dependent word tokenization method.

2 Related Work

The nature of our problem is similar to those reported in (Bekkerman and Gavish, 2011; Shen et al., 2011; Shen et al., 2012b; Yu et al., 2013b; Sun et al., 2014; Kozareva, 2015; ?), but with

¹The in-house datasets are from Rakuten USA, managed by Rakuten Ichiba, Japan’s largest e-commerce company.

²This dataset is from Rakuten Ichiba and is released under Rakuten Data Release program.

more pronounced data quality issues. However, the existing methods for noisy product classification have only been applied to English. Their efficacy for *moraic* and *agglutinative* languages such as Japanese remains unknown.

The work in Sun et al. (2014) emphasizes the use of simple classifiers in combination with large-scale manual efforts to reduce noise and imperfections from categorization outputs. While human intervention is important, we show how unsupervised topic models can substantially reduce such expensive efforts for product listings crawled in the wild. Further, unlike Sun et al. (2014), we adopt stronger baseline systems based on regularized linear models (Hastie et al., 2003; Zhang, 2004; Zou and Hastie, 2005).

A recent work from Pyo et al. (2016) emphasizes the use of recurrent neural networks for taxonomy categorization purposes. Although, they mention that RNNs render unlabeled pre-training of word vectors (Mikolov et al., 2013) unnecessary, in contrast, we show that training word embeddings on the whole set of three product title corpora improves performance for CNN models and opens up the possibility of leveraging other product corpora when available.

Shen et al. (2012b) advocate the use of algorithmic splitting of the taxonomy using graph theoretic latent group discovery to mitigate data imbalance problems at the leaf nodes. They use a combination of k-NN classifiers at the coarser level and SVMs (Cortes and Vapnik, 1995) classifiers at the leaf levels. Their SVMs solve much easier *k*-way multi-class categorization problems where $k \in \{3, 4, 5\}$ with much less data imbalance. We, however, have found that SVMs do not work well in scenarios where *k* is large and the data is imbalanced. Due to our high-dimensional feature spaces, we avoided k-NN classifiers that can cause

prohibitively long prediction times under arbitrary feature transformations (Manning et al., 2008; Cevahir and Murakami, 2016).

The use of a bi-level classification using k-NN and hierarchical clustering is incorporated in Cevahir and Murakami (2016)’s work, where they use nearest neighbor methods in addition to Deep Belief Networks (DBN) and Deep Auto Encoders (DAE) over both titles and descriptions of the Japanese product listing dataset. We show in Section 5.6, that using a tri-level cascade of GBT classifiers over titles, we significantly outperform the k-NN+DBN classifier on average.

3 Dataset Characteristics

We use two in-house datasets, named BU1 and BU2, one publicly available Amazon dataset (AMZ) (McAuley et al., 2015), and a Japanese product listing dataset named RAI (Cevahir and Murakami, 2016) (short for Rakuten Ichiba) for the experiments in this paper.

BU1 is categorized using human annotation efforts and rule-based automated systems. This leads to a high precision training set at the expense of coverage. On the other hand, for BU2, noisy taxonomy labels from external data vendors have been automatically mapped to an in-house taxonomy without any human error correction, resulting in a larger dataset at the cost of precision. BU2 also suffers from inconsistencies in regards to incomplete or malformed product titles and metadata arising out of errors in the web crawlers that vendors use to aggregate new listings. However, for BU2, the noise is distributed identically in the training and test sets, thus evaluation of the classifiers is not impeded by it.

The Japanese RAI dataset consists of 172,480,000 records split across 26,223 leaf nodes. The distribution of product listings in the leaf nodes is based on the popularity of certain product categories and is thus highly imbalanced. For instance, the top level “Sports & Outdoor” category has 2,565 leaf nodes, while the “Travel / Tours / Tickets” category has only 38. The RAI dataset has 35 categories at depth one (level-one categories) and 400 categories at depth two of the full taxonomy tree. The total depth of the tree varies from three to five levels.

The severity of data imbalance for BU2 is shown in Figure 2. The top-level “Home, Furniture and Patio” subtree that accounts for almost half of the BU2 dataset. Table 1 shows dataset

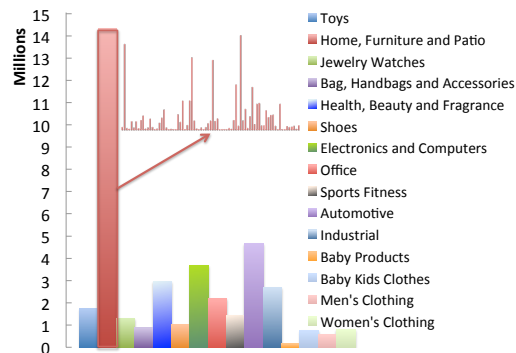


Figure 2: Top-level category distribution of 40 million deduplicated listings from an earlier Dec 2015 snapshot of BU2. Each category subtree is also imbalanced, as seen in exploded view of the “Home, Furniture, and Patio” category.

characteristics for the four different kinds of product datasets we use in our analyses. It lists the number of branches for the top-level taxonomy subtrees, the total number of branches ending at leaf nodes for which there are a non-zero number of listings and two important summary statistics that helps quantify the nature of imbalance. We first calculate the Pearson correlation coefficient (PCC) between the number of listings and branches in each of the top-level subtrees for each of the four datasets.

A perfectly balanced tree will have a PCC of 1.0. BU1 shows the most *benign* kind of imbalance with a PCC of 0.643. This confirms that the number of branches in the subtrees correlate well with the volume of listings. Both AMZ and RAI datasets show the highest branching factors in their taxonomies. For the AMZ dataset, it could be

Datasets	Subtrees	Branches	Listings	PCC	KL
BU1	16	1,146	12.1M	0.643	0.872
BU2	15	571	60M	0.209	0.715
AMZ	25	18,188	7.46M	0.269	1.654
RAI	35	26,223	172.5M	0.474	7.887

Table 1: Dataset properties on: total number of top-level category subtrees, branches and listings

due to the fact that the crawled taxonomy is different from Amazon’s internal catalog. The Rakuten Ichiba taxonomy has been incrementally adjusted to grow in size over several years by creating new branches to support newer and popular products. We observe that for RAI, AMZ and BU2 in particular, the number of branches in the subtrees do not correlate well with the volume of listings. This indicates a much higher level of imbalance.

We also compute the average Kullback-Leibler

(KL) divergence, $KL(p(\mathbf{x})|q(\mathbf{x}))$, (Cover and Thomas, 1991) between the empirical distribution over listings in branches for each subtree rooted in the nodes at depth one, $p(\mathbf{x})$, compared to a uniform distribution, $q(\mathbf{x})$. Here, the KL divergence acts as a measure of imbalance of the listing distribution and is indicative of the categorization performance that one may obtain on a dataset; high KL divergence leads to poorer categorization and vice-versa (see Section 5).

4 Gradient Boosted Trees and Convolutional Neural Networks

GBTs (Friedman, 2000) optimize a loss functional: $\mathcal{L} = E_y[L(y, F(\mathbf{x})|\mathbf{X})]$ where $F(\mathbf{x})$ can be a mathematically difficult to characterize function, such as a decision tree $f(\mathbf{x})$ over \mathbf{X} . The optimal value of the function is expressed as $F^*(\mathbf{x}) = \sum_{m=0}^M f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$, where $f_0(\mathbf{x}, \mathbf{a}, \mathbf{w})$ is the initial guess and $\{f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})\}_{m=1}^M$ are *additive boosts* on \mathbf{x} defined by the optimization method. The parameter \mathbf{a}_m of $f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$ denotes split points of predictor variables and \mathbf{w}_m denotes the boosting weights on the leaf nodes of the decision trees corresponding to the partitioned training set \mathbf{X}_j for region j . To compute $F^*(\mathbf{x})$, we need to calculate, for each boosting round m ,

$$\{\mathbf{a}_m, \mathbf{w}_m\} = \arg \min_{\mathbf{a}, \mathbf{w}} \sum_{i=1}^N L(y_i, F_m(\mathbf{x}_i)) \quad (1)$$

with $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + f_m(\mathbf{x}, \mathbf{a}_m, \mathbf{w}_m)$. This expression is indicative of a gradient descent step:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m (-g_m(\mathbf{x}_i)) \quad (2)$$

where ρ_m is the step length and $\left[\frac{\partial L(y, F(\mathbf{x}))}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)} = g_m(\mathbf{x}_i)$ being the search direction. To solve \mathbf{a}_m and \mathbf{w}_m , we make the basis functions $f_m(\mathbf{x}_i; \mathbf{a}, \mathbf{w})$ correlate most to $-g_m(\mathbf{x}_i)$, where the gradients are defined over the *training data* distribution. In particular, using Taylor series expansion, we can get closed form solutions for \mathbf{a}_m and \mathbf{w}_m – see Chen and Guestrin (2016) for details. It can be shown that $\mathbf{a}_m = \arg \min_{\mathbf{a}} \sum_{i=1}^N (-g_m(\mathbf{x}_i) - \rho_m f_m(\mathbf{x}_i, \mathbf{a}, \mathbf{w}_m))^2$ and $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \rho f_m(\mathbf{x}_i; \mathbf{a}_m, \mathbf{w}_m))$ which yields,

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m f_m(\mathbf{x}, \mathbf{a}_m, \mathbf{w}_m) \quad (3)$$

Each boosting round m updates the weights $w_{m,j}$ on the leaves and helps create a new tree

in the next iteration. The optimal selection of decision tree parameters is based on optimizing the $f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$ using a logistic loss. For GBTs, each decision tree is resistant to imbalance and outliers (Hastie et al., 2003), and $F(\mathbf{x})$ can approximate arbitrarily complex decision boundaries.

The convolutional neural network we use is based on the CNN architecture described in LeCun and Bengio (1995; Kim (2014) using the TensorFlow framework (Abadi and others, 2015). As in Kim (2014), we enhance the performance of “vanilla” CNNs (Fig. 3 **right**) using word embedding vectors (Mikolov et al., 2013) trained on the product titles from all datasets, without taxonomy labels. Context windows of width n , corresponding to n -grams and embedded in a 300 dimensional word embedding space, are convolved with L filters followed by rectified non-linear unit activation and a max-pooling operation over the set of all windows W . This operation results in a $L \times 1$ vector, which is then connected to a softmax output layer of dimension $K \times 1$, where K is the number of classes. Section A lists more details on parameters.

The CNN model tries to allocate as few filters to the context windows while balancing the constraints on the back-propagation of error residuals with regards to cross-entropy loss $\mathcal{L} = -\sum_{k=1}^K q_k \log p_k$, where p_k is the probability of a product title \mathbf{x} belonging to class k predicted by our model, and $q \in \{0, 1\}^K$ is a one-hot vector that represents the true label of title \mathbf{x} . This results in a higher predictive power for the CNNs, while still matching complex decision boundaries in a smoother fashion than GBTs. We note here that for all models, the predicted probabilities are *not* calibrated (Zadrozny and Elkan, 2002).

5 Experimental Setup and Results

We use Naïve Bayes (NB) (Ng and Jordan, 2001) similar to the approach described in Shen et al. (2012a) and Sun et al. (2014), and Logistic Regression (LogReg) classifiers with L_1 (Fan et al., 2008) and Elastic Net regularization, as robust baselines. Parameter setups for the various models and algorithms are mentioned in Section A.

5.1 Data Preprocessing

Product listing datasets in English – BU1 is exclusively comprised of product titles, hence, our features are primarily extracted from these titles. For AMZ and BU2, we additionally extract the list

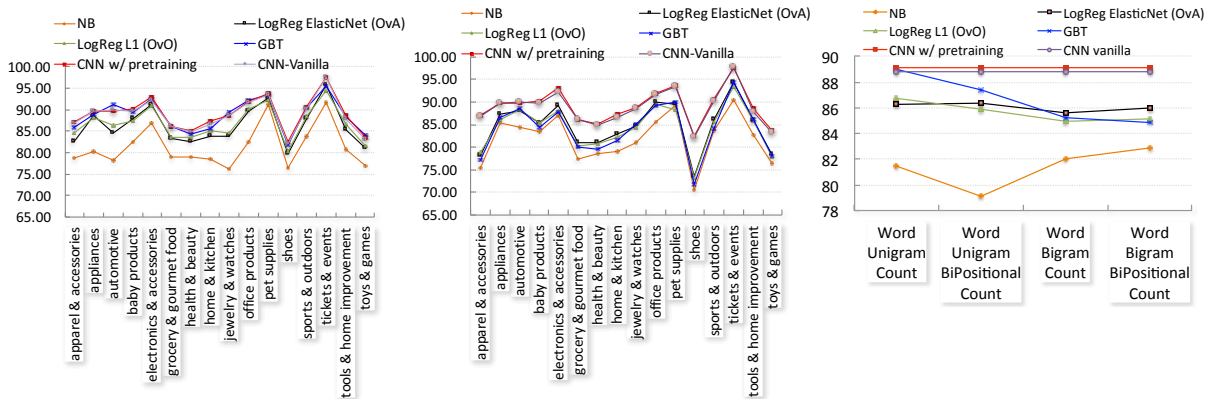


Figure 3: Classifier performance on BU1 test set. The CNN classifier has only one configuration and thus shows constant curves in all plots. **Left** figure shows prediction on 10% test set using word unigram count features; **middle** figure shows prediction on 10% test set using word bigram bi-positional count features; and the **right** figure shows mean micro-precision over different feature setups except CNNs. In all figures, “OvO” means “One vs. One” and “OvA” means “One vs All”.

price whenever available. For BU2, we also use the leaf node of any available *navigational breadcrumbs*. In order to decrease training and categorization run times, we employ a number of vocabulary filtering methods. Further, English stopwords and rare tokens that appear in 10 listings or less are then filtered out. This reduces vocabulary sizes by up to 50%, without a significant reduction in categorization performance. For CNNs, we replace numbers by the nominal form [NUM] and remove rare tokens. We also remove punctuations and then lowercase the resulting text. Parts of speech (POS) tagging using a generic tagger from Manning et al. (2014) trained on English text produced very noisy features, as is expected for out-of-domain tagging. Consequently, we do not use POS features due to the absence of a suitable training set for listings unlike that in Putthividhya and Hu (2011). For GBTs, we also experiment with title word expansion using nearest neighbors from Word2Vec model (Mikolov et al., 2013), for instance, to group words like “*t-shirts*”, “*tshirt*”, “*t-shirt*” in their respective equivalence classes, however, the overall results have not been better.

Product listing datasets in Japanese –CJK languages like Japanese lack white space between words. Hence, the first pre-processing step requires a specific Japanese tokenization tool to properly segment the words in the product titles.

For our experiments, we used the MeCab³ tokenizer trained using features that are augmented with in-house product keyword dictionaries. Romaji words written using Latin characters are sep-

arated from Kanji and Kana words. All brackets are normalized to square brackets and punctuations from non-numeric tokens are removed. We also use canonical normalization to change the code points of the resulting Japanese text into an NFKC normalized⁴ form, then remove anything outside of standard Japanese UTF-8 character ranges. Finally, the resulting text is lowercased.

Due to the size of the RAI dataset taxonomy tree, three groups of models are trained to classify new listings into one of 35 level-one categories, then one of 400 level-two categories, and, finally, the leaf node of the taxonomy tree. We have found this scheme to be working better for the RAI dataset than a bi-level scheme that we adopted for the other English datasets.

Applying GBTs on the Japanese dataset involved a bit more feature engineering. At the tokenized word-level, we use counts of word unigrams and word bi-grams. For character features, the product title is first normalized as discussed above. Consequently, character 2, 3, and 4-grams are extracted with their counts, where extractions include single spaces appearing at the end of word boundaries. Identification of the best set of feature combinations in this case has been performed during cross-validation.

5.2 Initial Experiments on BU1 dataset

Our initial experiments use unigram counts and three other features: word bigram counts, bi-positional unigram counts, and bi-positional bigram counts. Consider a title text “*120 gb hdd 5400rpm sata fdb 2 5 mobile*” from the “*Data*

³<https://sourceforge.net/projects/mecab/>

⁴<http://unicode.org/reports/tr15/>

storage” leaf node of the *Electronics* taxonomy subtree and another title text “acer aspire v7 582pg 6421 touchscreen ultrabook 15 6 full hd intel i5 4200u 8gb ram 120 gb hdd ssd nvidia geforce gt 720m” from the “Laptops and notebooks” leaf node. In such cases, we observe that merchants tend to place terms pertaining to storage device specifics in the front of product titles for “Data storage” and similar terms towards the end of the titles for “Laptops”. As such, we split the title length in half and augment word uni/bigrams with a left/right-half position.

This makes sense from a Naïve Bayes point of view, since terms like “120_gb”[Left_Half], “gb_hdd”[Left_Half], “120_gb”[Right_Half] and “gb_hdd”[Right_Half] de-correlates the feature space better, which is suitable for the naïve assumption in NB classification. This also helps in slightly better explanation of the class posteriors. These assumptions for NB are validated in the three figures: Fig. 3 **left**, Fig. 3 **middle** and Fig. 3 **right**. Word unigram count features perform strongly for all classifiers except NB, whereas bi-positional word bigram features helped only NB significantly.

Additionally, the micro-precision and F1 scores for CNNs and GBTs are significantly higher compared to other algorithms on word unigrams using paired *t*-test with a *p*-value < 0.0001. The performances of GBTs and LogReg L_1 classifiers deteriorate over the other feature sets as well. The bi-positional and bigram feature sets also do not produce any improvements for the AMZ dataset. Based on these initial results, we focus on word unigrams in all of our subsequent experiments.

5.3 Categorization Improvements with Navigational Breadcrumbs and List Prices on BU2 Dataset

BU2 is a challenging dataset in terms of class imbalance and noise and we sought to improve categorization performance using available meta-data. To start, we experiment with a smaller dataset consisting of $\approx 500,000$ deduplicated listings under the “Women’s Clothing” taxonomy subtree, extracted from our Dec 2015 snapshot of 40 million records. Then we train and test against ≈ 2.85 million deduplicated “Women’s Clothing” listings from the Feb 2016 snapshot of BU2. In all experiments, 10% of the data is used as test set. The womens clothing category had been chosen due to the importance of the category from a business

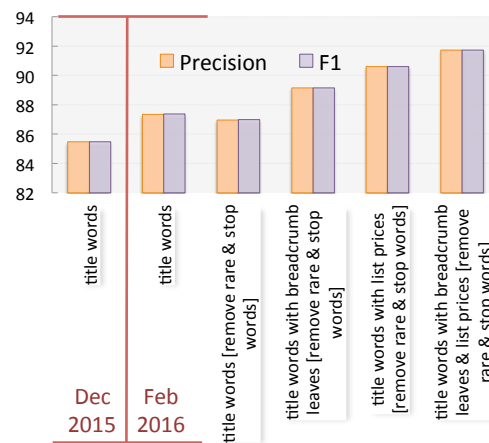


Figure 4: Improvements in micro-precision and F1 for GBTs on BU2 dataset for “Women’s Clothing” subtree

standpoint, which provided early access to listings in this category. Further, data distributions remain the same in the two snapshots and the Feb 2016 snapshot consists of listings in addition to those for the Dec 2015 snapshot.

The first noteworthy fact in Fig. 4 is that the micro-precision and F1 of the GBTs substantially improve after increasing the size of the dataset. Further, stop words and rare words filtering decrease precision and F1 by less than 1%, despite halving the feature space. The addition of navigational leaf nodes and list prices prove advantageous, with both features independently boosting performance and raising micro-precision and F1 to over 90%. Despite finding similar gains in categorization performance for other top-level subtrees by using these meta features, we needed a system to filter mis-categorized listings from our training data as well.

5.4 Noise Analysis of BU2 Dataset using Correspondence LDA for Text

The BU2 dataset has the noisiest ground-truth labels, as incorrect labels have been assigned to product listings. However, since the manual verification of millions of listings is infeasible, using some proxy for ground truth is a viable alternative that has previously produced encouraging results (Shen et al., 2012b). We next describe how resorting to unsupervised topic models helped to detect and remove incorrect listings.

As shown in Fig. 8, categorization performance for the “Shoes” taxonomy subtree is over 25 points below the “Women’s Clothing” category. Such a large difference could be caused by incorrect assignments of listings to the correct cat-

hardcover guide design handbook international health business social law	heart diamond pearl 40 ring sterling chain charm 39 14k 47	paperback book history god home und soul stories journey bible der
vhs world series time king war ball house trek christmas space	i c love day e night u single lady child woman uk good deluxe park	set 20 100 24 case kit oz 30 hand drive body wall digital ft 48 spray paper
license standard symantec system service cisco support year	dvd jazz media mixed country artists vol play music product pop	de calendar la american disc el 2013 art 2009 ray 2012 compact

Figure 5: Selection of most probable words under the latent “noise” topics over listings in “*Shoes*” subtree. Human annotators inspect whether such sets of words belong to a Shoes topic or not.

egories. However, unlike Sun et al. (2014), as there are over 3.4 million “*Shoes*” listings in the BU2 dataset, a manual analysis to detect noisy labels is infeasible. To address this problem, we compute $p(\mathbf{x})$ over latent topics z_k , and automatically annotate the most probable words over each topic.

We choose our CorrMMLDA model (Das et al., 2011) to discover the latent topical structure of the listings in the “*Shoes*” category because of two reasons. Firstly, the model is a natural choice for our scenario since it is intuitive to assume that store and brand names are distributions over words in titles. This is illustrated in the graphical model in Fig. 7, where the store “*Saks Fifth Avenue*” and the brand “*Joie*” are represented as words $w_{d,m}$ in the M plate of listing d and are distributions over the words in the product title “*Joie kidmore Embossed slipon sneakers*” represented as words $w_{d,n}$ in the N plate of the same listing d . The title words are in turn distributions over the latent topics z_d for listing $d \in \{1..D\}$.

Secondly, the CorrMMLDA model has been shown to exhibit lower held-out perplexities indicative of improved topic quality. The reason behind the lower perplexity stems from the following observations: Using the notation in Das et al. (2011), we denote the free parameters of the variational distributions over words in brand and store names, say $\lambda_{d,m,n}$, as multinomials over words in the titles and those over words in the title, say $\phi_{d,n,k}$, as multinomials over latent topics z_d . It is easy to see that the posterior over the topic $z_{d,k}$ for each $w_{d,m}$ of brand and store names, is dependent on λ and ϕ through $\sum_{n=1}^{N_d} \lambda_{d,m,n} \times \phi_{d,n,k}$. This means that if a certain topic $z_d = j$ generates all words in the title, i.e., $\phi_{d,n,j} > 0$, then

oxford burgundy plain espresso wing madison	Oxfords > Men’s Shoes > Shoes
pr boots work mens blk composite	Boots > Men’s Shoes > Shoes
tan polyurethane life saddle stride bed	Flats > Women’s Shoes > Shoes
13 reaction york ak driving steven	Sneakers & Athletic Shoes > Men’s Shoes > Shoes
original rose bone mark lizard copper	Climbing > Men’s Shoes > Shoes

Figure 6: Interpretation of latent topics using predictions from a GBT classifier. The topics here do not include those in Fig. 5, but are all from Feb 2016 snapshot of the BU2 dataset.

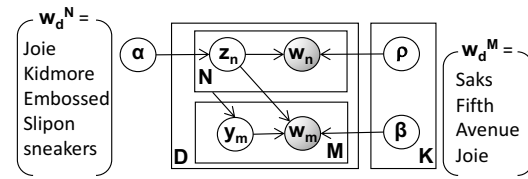


Figure 7: Correspondence MMLDA model.

only that topic also generates the brand and store names thereby increasing likelihood of fit and reducing perplexity. The other topics $z_d \neq j$ do not contribute towards explaining the topical structure of the listing d .

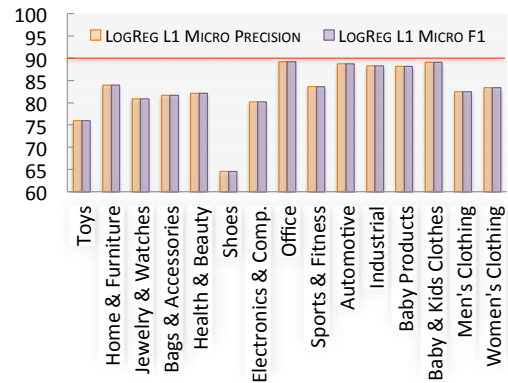


Figure 8: Micro-precision and F1 across fifteen top-level categories on 10% (4 million listings) of Dec 2015 BU2 snapshot.

We train the CorrMMLDA model with $K=100$ latent topics. A sample of nine latent topics and their most probable words shown in Fig. 5 demonstrates that topics outside of the “*Shoes*” domain can be manually identified, while reducing human annotation efforts from 3.4 million records to one hundred. We choose $K = 100$ since it is roughly twice the number of branches for the Shoes subtree. This choice provides modeling flexibility while respecting the number of ground

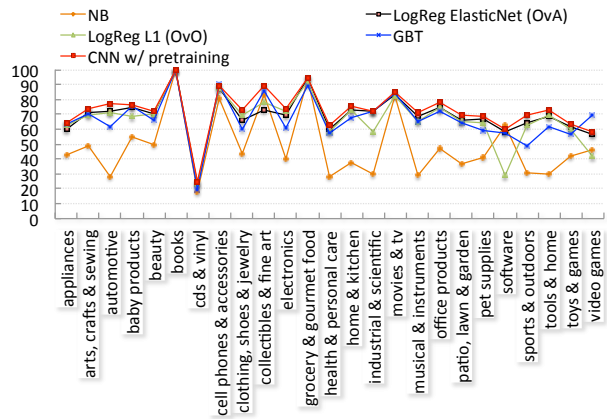
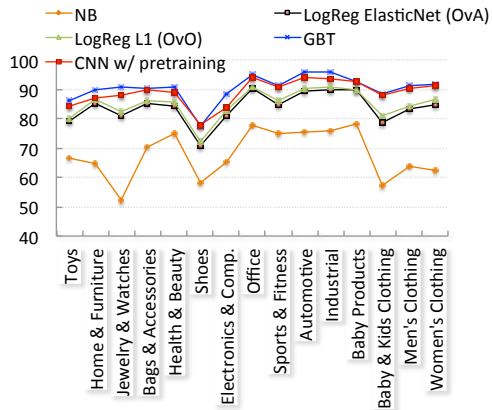


Figure 9: Micro-precision on 10% of BU2 across categories (see Sect. 5.4)

Figure 10: Micro-precision on 10% of AMZ across categories

Dataset	NB	LogReg ElasticNet	LogReg L1	GBT	CNN w pretraining	Mean KL	$\log(N/B)$
BU1	81.45	86.30	86.75	89.03*	89.12*	0.872	9.27
BU2	68.21	84.29	85.01	90.63*	88.67	0.715	11.54
AMZ	49.01	69.39	66.65	67.17	72.66*	1.654	6.02

Table 2: Mean micro-precision on 10% test set from BU1, BU2 and AMZ English datasets

truth classes.

We next run a list of the most probable six words, the average length of a “*Shoes*” listing’s title, from each latent topic through our GBT classifier trained on the full, noisy data, *but without considering any metadata*, due to bag-of-words nature of the topic descriptions. As shown in the bottom two rows in Fig. 6, categories mismatching their topics are manually labeled as ambiguous. As a final validation, we uniformly sampled a hundred listings from each ambiguous topic detected by the model. Manual inspections revealed numerous listings from merchants not selling shoes are wrongly cataloged in the “*Shoes*” subtree due to vendor’s error. To this end, we remove listings corresponding to such “*out-of-category*” merchants from all top-level categories.

Thus, by manually inspecting $K \times 6$ most probable words from the $K=100$ topics and $J \times 100$ listings, where $J \ll K$, instead of 3.4 million, a few annotators accomplished in hours what would have taken hundreds of annotators several months according to the estimates in Sun et al. (2014).

5.5 Results on BU2 and AMZ Datasets

In section 5.2, we have shown the efficacy of word unigram features on the BU1 dataset. Figure 8 shows that LogReg with L_1 regularization (Yu et al., 2013b; Yu et al., 2013a) initially achieves 83% mean micro-precision and F1 on the initial BU2 dataset. This falls short of our expectation of achieving an overall 90% precision (red line in

Fig. 8), but forms a robust baseline for our subsequent experiments with the AMZ and the cleaned BU2 datasets. We additionally use the list price and the navigational breadcrumb leaf nodes for the BU2 dataset and, when available, the list price for the AMZ dataset.

Overall, Naïve Bayes, being an overly simplified generative model, generalizes very poorly on all datasets (see Figs. 3, 9 and 10). A possible option to improve NB’s performance is to use sub-sampling techniques as described in Chawla et al. (2002); however, sub-sampling can have its own problems for when dealing with product datasets (Sun et al., 2014).

From Table 2, we observe that most classifiers tend to perform well when $\log(N/B)$ is relatively high. The N in the previous ratio is the total number of listings and B is the total number of categories. Figures with a * are statistically better than other non-starred ones in the same row except the last two columns. From Fig. 9 and Table 2, it is clear that GBTs are better on BU2.

We also experiment with CNNs augmented to use meta-data while respecting the convolutional constraints on title text, however, the performance improved only marginally. It is not immediately clear why all the classifiers suffer on the “*CDs and Vinyl*” category, which has more than 500 branches – see Fig. 10. The AMZ dataset also suffers from novel cases of data imbalance. For instance, most of the listings in “*Books*” and “*Gro-*

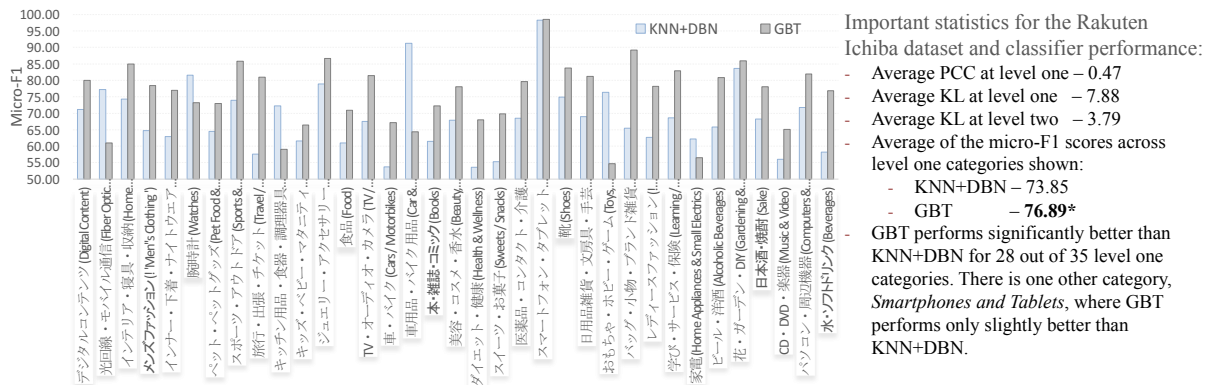


Figure 11: Comparison of GBTs versus the method from Cevahir and Murakami (2016) on a 10% test set from the Rakuten Ichiba Japanese product listing dataset.

cery” are in one branch, with most other branches containing less than 10 listings. In summary, from both Figs. 9 and 10, we observe that GBTs and CNNs with pre-training perform best even in extreme data imbalance. It is possible that GBTs need finer parameter tuning per top-level subtree for datasets resembling AMZ.

5.6 Results on Rakuten Ichiba Dataset

In this section, we report our findings on the efficacy of GBTs vis-a-vis another hybrid nearest neighbor and deep learning based method from Cevahir and Murakami (2016). Our decision to employ a tri-level classifier cascade, instead of the bi-level one used for the other datasets, stems from our observations of the KL divergence values (see Section 3 and Table 1) at the first and second level depths of the RAI taxonomy tree. Moving from the first level down to the second decreases the KL divergence by more than 50%. We thus expect GBTs to perform better due to this reduced imbalance. We also cross-validated this assumption on some popular categories, such as “Clothing”.

From Fig. 11 and the statistics noted therein, we observe that, on average, GBTs outperform the KNN+DBN model from Cevahir and Murakami (2016) by 3 percentage points across all top level categories, which is statistically significant under a paired t -test with $p < 0.0001$. As with previous experiments, only a common best parameter configuration has been set for GBTs, without resorting to time consuming cross-validation across all categories. For the 29 categories on which GBTs do better, the mean of the absolute percentage improvement is 11.78, with a standard deviation of 5.07. Also, it has been observed that GBTs **significantly** outperform KNN+DBN in 28 of those categories.

The comparison in Fig. 11 is more holistic. Un-

like the top level categorization scores obtained in Figs. 3, 9 and 10, the scores in Fig. 11 have been obtained by categorizing each test example through the entire cascade of hierarchical models for two classifiers. Even with this setting, the performance of GBTs is significantly better.

6 Conclusion

Large-scale taxonomy categorization with noisy and imbalanced data is a challenging task. We demonstrate deep learning and gradient tree boosting models with operational robustness in real industrial settings for e-commerce catalogs with several millions of items. We summarize **our contributions** as follows: 1) We conclude that GBTs and CNNs can be used as new state-of-the-art baselines for product taxonomy categorization problems, **regardless of the language used**; 2) We quantify the nature of imbalance for different product datasets in terms of distributional divergence and correlate that to prediction performance; 3) We also show evidence to suggest that words from product titles, together with leaf nodes from navigational breadcrumbs and list prices, when available, can boost categorization performance significantly on all the product datasets. Finally, 4) we showcase a novel use of topic models with minimal human intervention to clean large amounts of noise particularly when the source of noise cannot be controlled. This is unlike any experiment reported in previous publications on product categorization. Automatic topic labeling for a given category with a pre-trained classifier from another dataset can help create an initial taxonomy over listings for which none exist. A major benefit of this approach is that it reduces manual efforts on initial taxonomy creation.

References

- Martin Abadi et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems.
- Ron Bekkerman and Matan Gavish. 2011. High-precision phrase-based document classification on a modern scale. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 231–239, New York, NY, USA. ACM.
- Ali Cevahir and Koji Murakami. 2016. Large-scale multi-class and hierarchical product categorization for an e-commerce giant. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 525–535, Osaka, Japan, December. The COLING 2016 Organizing Committee.
- Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June.
- Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794.
- Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September.
- Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA.
- Pradipto Das, Rohini Srihari, and Yun Fu. 2011. Simultaneous joint and conditional modeling of documents tagged from two perspectives. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 1353–1362, New York, NY, USA. ACM.
- Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.
- Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.
- Venkatesh Ganti, Arnd Christian König, and Xiao Li. 2010. Precomputing search features for fast and accurate query classification. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10)*. Society for Artificial Intelligence and Statistics.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2003. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, August.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1329–1333.
- Y. LeCun and Y. Bengio. 1995. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15*, pages 785–794, New York, NY, USA. ACM.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.
- Andrew Ng and Michael Jordan. 2001. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14.

- Duangmanee (Pew) Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11*, pages 1557–1567, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Hyuna Pyo, Jung-Woo Ha, and Jeonghee Kim. 2016. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, New York, NY, USA. ACM.
- Dan Shen, Jean David Ruvini, Manas Somaiya, and Neel Sundaresan. 2011. Item categorization in the e-commerce domain. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 1921–1924, New York, NY, USA. ACM.
- Dan Shen, Jean-David Ruvini, Rajyashree Mukherjee, and Neel Sundaresan. 2012a. A study of smoothing algorithms for item categorization on e-commerce sites. *Neurocomput.*, 92:54–60, September.
- Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012b. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 595–604, New York, NY, USA. ACM.
- Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. 2014. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proc. VLDB Endow.*, 7(13), August.
- Hsiang-Fu Yu, Chia-Hua Ho, Yu-Chin Juan, and Chih-Jen Lin. 2013a. LibShortText: A Library for Short-text Classification and Analysis. Technical report, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan.
- Hsiang-Fu Yu, Chia hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih jen Lin. 2013b. Product title classification versus text classification. Technical report, UTexas, Austin; NTU; EBay.
- Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02*, pages 694–699, New York, NY, USA. ACM.
- Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 2004: Proceedings of the 21st International Conference on Machine Learning*, pages 919–926.
- Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

A Supplemental Materials: Model Parameters

In this paper, the baseline classifiers comprise of Naïve Bayes (NB) (Ng and Jordan, 2001) similar to the approach described in Shen et al. (2012a) and Sun et al. (2014), and Logistic Regression (LogReg) classifiers with L_1 (Fan et al., 2008) and Elastic Net regularization. The objective functions of both GBTs and CNNs involve L_2 regularizers over the set of parameters. Our development set for parameter tuning is generated by randomly selecting 10% of the listings under the “*apparel / clothing*” categories. The optimized parameters obtained from this scaled-down configuration is then extended to all other classifiers to reduce experimentation time.

For parameter tuning, we set a linear combination of 15% L_1 regularization and 85% L_2 regularization for Elastic Net. For GBTs (Chen and Guestrin, 2016) on both English and Japanese data, we limit each decision tree growth to a maximum depth of 500 and the number of boosting rounds is set to 50. Additionally, for leaf node weights, we use L_2 regularization with a regularization constant of 0.5. For GBTs on English data, the initial learning rate is 0.2. For GBTs on Japanese data, the initial learning rate is assigned a value of 0.05 .

For CNNs, we use context window widths of sizes 1, 3, 4, 5 for four convolution filters, a batch size of 1024 and an embedding dimension of 300. The parameters for the embeddings are non-static. The convolutional filters are initialized with Xavier initialization (Glorot and Bengio, 2010). We use mini-batch stochastic gradient descent with Adam optimizer (Kingma and Ba, 2014) to perform parameter optimization.

LogReg classifiers and CNN need data to be normalized along each dimension, which is not needed for NB and GBT.

Recognizing Insufficiently Supported Arguments in Argumentative Essays

Christian Stab[†] and Iryna Gurevych^{†‡}

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)

Department of Computer Science, Technische Universität Darmstadt

[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)

German Institute for Educational Research

www.ukp.tu-darmstadt.de

Abstract

In this paper, we propose a new task for assessing the quality of natural language arguments. The premises of a well-reasoned argument should provide enough evidence for accepting or rejecting its claim. Although this criterion, known as sufficiency, is widely adopted in argumentation theory, there are no empirical studies on its applicability to real arguments. In this work, we show that human annotators substantially agree on the sufficiency criterion and introduce a novel annotated corpus. Furthermore, we experiment with feature-rich SVMs and convolutional neural networks and achieve 84% accuracy for automatically identifying insufficiently supported arguments. The final corpus as well as the annotation guideline are freely available for encouraging future research on argument quality.¹

1 Introduction

Argumentation is an omnipresent routine and an integral part of our daily verbal communication. It is a verbal activity that aims at increasing or decreasing the plausibility of a controversial standpoint (van Eemeren et al., 1996, p. 5). Well-reasoned arguments of high quality are not only important for making thoughtful decisions and persuading a particular audience but also play a major role for drawing widely accepted conclusions. Computational argumentation is a recent research field in natural language processing that focuses on the analysis of arguments in natural language texts. Novel advances have a broad applica-

tion potential in various areas like debating technologies (Levy et al., 2014; Rinott et al., 2015), policy making (Sardianos et al., 2015), information retrieval (Carstens and Toni, 2015), and legal decision support (Mochales-Palau and Moens, 2009). Recently, computational argumentation is receiving increasing interest in *intelligent writing assistance* (Song et al., 2014; Stab et al., 2014) since it enables *argumentative writing support* systems that provide tailored feedback about arguments in student essays.

Most of the existing approaches in computational argumentation consider argumentation as discourse structures and focus on the identification of arguments in natural language texts. For instance, existing approaches classify text units as argumentative or non-argumentative (Moens et al., 2007), recognize argument components such as *claims* or *premises* at the sentence-level (Mochales-Palau and Moens, 2009; Kwon et al., 2007; Eckle-Kohler et al., 2015) or clause-level (Levy et al., 2014; Sardianos et al., 2015), or identify argument structures by classifying pairs of argument components (Stab and Gurevych, 2014). However, these approaches are of limited use for argumentative writing support systems since they do not recognize the weak points of arguments.

Despite the comprehensive theoretical framework on argument quality in logic and argumentation theory (van Eemeren et al., 1996; Damer, 2009), there are only few computational approaches that focus on the assessment of arguments in natural language texts. These existing approaches either identify undisputed arguments in online communities (Cabrio and Villata, 2012), assess the persuasiveness of arguments (Wei et al., 2016), compare and rank arguments regarding their convincingness (Habernal and Gurevych, 2016b), or summarize the argumentation strength

¹<https://www.ukp.tu-darmstadt.de/data/argumentation-mining>

of an entire essay in a single holistic score (Persing and Ng, 2015). Our approach is based on the theoretical framework proposed by Johnson and Blair (2006). In particular, we focus on the *sufficiency criterion* that an argument fulfills if its premises provide enough evidence for accepting or rejecting the claim. The following example argument illustrates a violation of the sufficiency criterion:

Example 1: “*It is an undeniable fact that tourism harms the natural habitats of the destination countries. As Australia’s Great Barrier Reef has shown, the visitors cause immense destruction by breaking corals as souvenirs, throwing boat anchors or dropping fuel and other sorts of pollution.*”

The premise of this argument represents a particular example (second sentence) that supports a general claim in the first sentence. The argument is a generalization from one sample to the general case. However, a single sample is not enough to support the general case. Therefore, the argument does not comply with the sufficiency criterion.

Example 2: “*Cloning will be beneficial for people who are in need of organ transplants. Cloned organs will match perfectly to the blood group and tissue of patients since they can be raised from cloned stem cells of the patient. In addition, it shortens the healing process.*”

Example 2 illustrates a sufficiently supported argument. It is reasonable to accept that transplantation patients will benefit from cloning if it enables a better match and an accelerated healing process.

Our primary motivation is to create an argument analysis method for argumentative writing support systems that classifies an argument as *sufficient* if its premises provide enough evidence for accepting its claim (example 2) or as *insufficient* if its premises do not provide enough evidence (example 1). Therefore, our first research question is whether human annotators can reliably apply the sufficiency criterion to real arguments and if it is possible to create annotated data of high quality. The second research question addresses the automatic recognition of insufficiently supported arguments. We investigate if, and how accurately, insufficiently supported arguments can be identified by computational techniques.

The contribution of this paper is threefold: first, we investigate to what extent human annotators agree on the sufficiency criterion. We present the results of an annotation study with three annotators and show that our annotation guideline successfully guides annotators to substantial agreement. Second, we show that insufficiently supported arguments can be identified with high accuracy using convolutional neural networks (CNN). The experimental results show that a CNN significantly outperforms several challenging baselines and manually created features. Third, we introduce a novel corpus for studying the quality of arguments.

2 Related Work

Previous works in computational argumentation focused primarily on approaches for *argument mining*. These include, for example, methods for the identification of arguments in legal texts (Moens et al., 2007), news articles (Eckle-Kohler et al., 2015; Sardianos et al., 2015), or user-generated web discourse (Habernal and Gurevych, 2016a). Other approaches address the classification of argument components into claims and premises (Mochales-Palau and Moens, 2009), supporting and opposing claims (Kwon et al., 2007), or backings, rebuttals and refutations (Habernal and Gurevych, 2016a). Levy et al. (2014) recognize context-dependent claims and Rinott et al. (2015) retrieve several types of evidence from Wikipedia. Approaches for identifying the structure of arguments recognize argumentative relations between argument components using context-free grammars (Mochales-Palau and Moens, 2009), pair classification (Stab and Gurevych, 2014), or maximum spanning trees (Peldszus and Stede, 2015). However, none of these approaches consider the quality of arguments.

Similarly, most existing corpora in computational argumentation are only annotated with argument components (Habernal and Gurevych, 2016a; Aharoni et al., 2014; Mochales-Palau and Moens, 2009) or argument structures (Reed et al., 2008; Stab and Gurevych, 2014; Peldszus and Stede, 2015) and do not include annotations of argumentative quality issues. Other resources in the field contain arguments annotated with different properties such as emotions and sarcasm (Walker et al., 2012), the type of reasoning (Reed et al.,

2008) or the stance on a topic (Somasundaran and Wiebe, 2009). However, there is no corpus of arguments annotated with the sufficiency criterion.

Currently there are only few approaches that focus on the automatic assessment of argument quality. Cabrio and Villata (2012) employed textual entailment for identifying undisputed arguments in online discussions. They built a graph that represents attack and support relations between arguments and applied the abstract argumentation framework (Dung, 1995) for identifying accepted arguments. Although their approach is capable of finding undisputed arguments among a given set of arguments, it does not answer why a specific argument is of inferior quality than another argument. Thus, their approach is of limited use for guiding students since it does not pinpoint particular weaknesses of arguments.

Park and Cardie (2014) proposed an approach for classifying propositions as verifiable (experiential and non-experiential) or unverifiable. Their best approach based on a support vector machine achieves a macro F1 score of .690. Although the verifiability of propositions enables to determine appropriate types of support, it does not answer if an argument is sufficiently supported or not.

Persing and Ng (2015) introduced an approach for recognizing the argumentation strength of an essay. They found that pos n-grams, prompt adherence features, and predicted argument components perform best. However, their model determines a single holistic score that summarizes the argumentation quality of the entire essay. Consequently, it does not provide formative feedback that guides students to improve their arguments.

Recently, researchers proposed approaches for automatically assessing the persuasiveness of arguments. For instance, Wei et al. (2016) proposed an approach for ranking user comments taken from online fora and found that argumentation related features are effective for this task. Cano-Basave and He (2016) ranked speakers in political debates by using semantic frames which indicate persuasive argumentation features, and Habernal and Gurevych (2016b) compared the convincingness of argument pairs using feature-rich SVMs and bidirectional LSTMs. However, the persuasiveness score of an argument is only of limited use for argumentative writing support, since it summarizes various quality criteria and does not explain why an argument is weak.

3 Argument Quality: Theoretical Background

An argument consists of several *argument components*. It includes a claim and one or more premises. The *claim* (also called *conclusion*) is a controversial statement and the central component of an argument. The *premises* constitute the reasons for believing the claim to be true or false (Damer, 2009, p. 14). Assessing the quality of arguments is a complex task since arguments in natural language are hardly ever in a standardized form (Damer, 2009; Govier, 2010). Moreover, argument quality is a product of many different criteria (Johnson and Blair, 2006). The quality of an argument depends, for instance, on its lexical clarity and phrasing (representation), the level of trust that the audience has in the arguer (ethos), and the emotions and values appealed by the argument (pathos). The *logical quality* of arguments (logos) is, however, independent of all other merits, defects and external influence factors (Johnson and Blair, 2006, p. 50). Certainly, external factors or the presentation style can have a strong influence on the persuasive power of arguments. However, these factors can at most masquerade an illogical argument but not improve its logical quality. Therefore, the logical quality is most suitable for assessing the (intrinsic) quality of arguments and for providing feedback about written arguments respectively.

Traditionally, there are two different perspectives on the logical quality of arguments: (i) the formal logic perspective and (ii) the informal logic perspective. The objective of *formal logic approaches* is to distinguish deductively valid arguments from invalid arguments (van Eemeren et al., 1996, chapter 1.2), i.e. to recognize if the claim of an argument follows necessarily from its premises. However, formal logic approaches cannot be applied to everyday arguments since the vast majority of arguments do not follow deductive inference rules (Damer, 2009; van Eemeren et al., 1996).

Informal logic aims at developing theoretical frameworks for analyzing arguments in ordinary natural language (Groarke, 2015). These include, for example, *fallacy theories* which focus on determining particular argumentative mistakes that can be observed with a marked degree of frequency. Current theories list various forms of fallacious arguments. For instance, the framework proposed by Damer (2009) describes 61 different fallacy

types. However, fallacy theories are not appropriate for recognizing logically good arguments (van Eemeren et al., 1996, p. 178) since it is unknown if all fallacies are already known. To overcome this limitation, Johnson and Blair (2006) proposed three binary criteria, known as RAS-criteria, that a logically good argument needs to fulfill:

- *Relevance*: An argument fulfills the relevance criterion, if all of its premises count in favor of the truth (or falsity) of the claim.
- *Acceptability*: An argument fulfills the acceptability criterion if its premises represent undisputed common knowledge or facts.
- *Sufficiency*: An argument complies with the sufficiency criterion if its premises provide enough evidence for accepting or rejecting the claim.

The relevance criterion addresses the relation between each premise and the claim whereas the acceptability criterion focuses on the truthfulness of each individual premise. Both need to be evaluated independently for each premise of the argument. The sufficiency criterion addresses the premises of an argument together. It is fulfilled if the relevant premises of an argument are enough for justifying (or rejecting) the claim. The sufficiency criterion presupposes a non-empty set of relevant premises. However, an argument can violate the relevance criterion and comply with the sufficiency criterion at the same time. For instance, an argument can have several relevant premises that are sufficient for accepting the claim and additional premises that are not relevant to the claim. This also implies that a sufficient argument has a non-empty set of relevant premises but it is unknown if all premises of a sufficient argument are relevant to the claim.

In contrast to fallacy theories, the RAS-criteria enable to distinguish good from bad arguments with respect to logical quality since each argument that complies with all three criteria is a logically good one (Govier, 2010; Johnson and Blair, 2006). Moreover, the RAS-criteria attribute a particular defect to the relation between individual premises and the claim (relevance), the truthfulness of individual premises (acceptability), or the premises considered together (sufficiency). Therefore, they enable purposeful feedback for resolving particular defects of weak arguments and are well suited for argumentative writing support systems.

4 Corpus Creation

We conducted our annotation on a corpus of 402 argumentative essays that has been previously annotated with argumentation structures (Stab and Gurevych, 2016). By analyzing the annotated argumentation structures, we found that each body paragraph contains at least one argument and only 4.3% of all body paragraphs include several arguments, i.e. claims supported by premises. Therefore, we considered each body paragraph as an individual argument. This approximation has additional practical advantages for the identification of insufficiently supported arguments since it does not require the identification of argumentation structures in advance and prevents potential error propagation. Following this procedure, we extracted 1,029 arguments with an average length of 94.6 tokens and 4.5 sentences per argument.

4.1 Annotation Study

Three non-native annotators with excellent English proficiency independently annotated the arguments as sufficient or insufficient. We used 64 arguments from the corpus for elaborating the annotation guideline and 20 arguments for collaborative training sessions with the annotators. In these sessions, all three annotators collaboratively analyzed arguments for resolving disagreements and obtaining a common understanding of the annotation guideline. For the actual annotation task, we used the freely available brat rapid annotation tool (Stenetorp et al., 2012).

4.1.1 Inter-Annotator Agreement

All three annotators independently annotated an evaluation set of 433 arguments. We evaluated the agreement between the annotators using several inter-annotator agreement measures implemented in DKPro Agreement (Meyer et al., 2014). We used observed agreement and the two chance-corrected measures Fleiss' κ (Fleiss, 1971) and Krippendorff's α with nominal distance function (Krippendorff, 1980). The three annotators agreed on 91.07% of all 433 arguments (observed agreement). The chance-corrected agreement scores of $\kappa = .7672$ and $\alpha = .7673$ show substantial agreement between the annotators which allows "tentative conclusions" (Krippendorff, 1980). Therefore, we conclude that human annotators can reliably identify insufficiently supported arguments in argumentative essays.

4.1.2 Analysis of Disagreements

In order to identify the reasons for the disagreements, we manually investigated all arguments on which the annotators disagreed. We found that a high proportion of these arguments include modal verbs in their claims. The following example illustrates such an argument:

“Watching television too often can have a negative effect on communication abilities. For instance, children often prefer watching cartoons or movies instead of meeting their classmates and thus they will not learn how to communicate properly.”

Due to the modal verb “*can*” in the claim of this argument (first sentence), it is sufficient to provide one specific example as premise. However, annotators tend to overlook modal verbs and over-hastily annotate these arguments as insufficient.

The second most frequent cause of disagreements is due to the length of the arguments. In particular, one annotator annotated remarkably fewer arguments as insufficient. These arguments exhibit a comparatively large number of premises. This indicates that longer arguments are more likely to be perceived as sufficient than shorter arguments.

We also observed that several disagreements are due to hard cases. For instance, assessing the sufficiency of the following argument depends on the subjective interpretation of the undetermined quantification “*many*” in the claim:

“Living in big cities provides many opportunities. First of all, it will be easier to find a job in a city. Also there are various bars and clubs where you can meet new people. Above all there are shopping malls and cinemas for spending your free time.”

We also found that annotators do not agree on arguments including terms such as “*some*”, “*various*”, or “*large number*”. Thus, extending the annotation guideline with an explanation of how to handle modal verbs, the number of premises and undetermined qualifiers could further improve the agreement between the annotators in future annotation studies.

4.2 Creation of the Final Corpus

We merged the annotations of the three annotators on the evaluation set using majority voting. The remaining arguments have been annotated by the two annotators with the highest pairwise agreement on the evaluation set ($\alpha = .815$). Disagreements on the remaining arguments have been manually resolved in discussions among the two annotators. Table 1 shows an overview of the corpus.

size	
tokens	97,370
sentences	4,593
arguments	1,029
class distribution	
sufficient	681 (66.2%)
insufficient	348 (33.8%)

Table 1: Size of the final corpus and class distribution of sufficiency annotations.

The class distribution is skewed towards sufficiently supported arguments. However, the proportion of 33.8% insufficiently supported arguments indicates that students frequently do not support their claims with sufficient evidence.

5 Experiments

We consider the identification of insufficiently supported arguments as a binary classification task and label each body paragraph as *sufficient* or *insufficient*. For preventing errors in model assessment due to a particular data splitting (Krstajic et al., 2014), we used a repeated 5-fold cross-validation setup and ensured that arguments from the same essay are not distributed over the train, test and development sets. We repeated the cross-validation 20 times which yields a total of 100 folds. As evaluation scores, we used accuracy and macro F1 score as well as the F1 score, precision and recall of the class “insufficient”. Whereas the precision indicates the performance of the model to identify arguments that are really in need of revision, recall shows how well the model recognizes all insufficiently supported arguments in an essay. All evaluation scores are reported as average including the standard deviation over the 100 folds. In order to determine the macro F1 score, we employ macro-averaging as proposed by Sokolova and Lapalme (2009, p. 430). For model selection and hyperparameter tuning, we randomly sampled 10% of the training set of each

fold as a development set. For significance testing, we employ Wilcoxon signed-rank test on macro F1 scores with a significance level of $\alpha = .005$.

We employ several models from the DKPro Framework (Eckart de Castilho and Gurevych, 2014) for preprocessing. We use the language tool segmenter² for tokenization and sentence splitting. We employ the Stanford parser (Klein and Manning, 2003) and named entity recognizer (Finkel et al., 2005) for constituency parsing and recognizing organizations, persons and locations. Note that only the model described in Section 5.2 requires all preprocessing steps. All other models use only the tokenization of the language tool segmenter.

5.1 Baselines

For our experiments, we use the following two baselines: First, we employ a majority baseline that classifies each argument as sufficient. Second, we use a support vector machine with polynomial kernel implemented in the Weka framework (Hall et al., 2009). We employ the 4,000 most frequent lowercased words as binary features and refer to this model as SVM-bow.

5.2 Manually Created Features (SVM)

Our first system is based on manually created features. As a learner, we use the same support vector machine as for SVM-bow. For feature extraction and experimentation, we use the DKPro TC text classification framework (Daxenberger et al., 2014). We tried various features which have been used previously for assessing the quality or the persuasiveness of arguments (cf. Section 2). For instance, we experimented with argument structures (Stab and Gurevych, 2014), transitional phrases (Persing and Ng, 2015), semantic roles (Das et al., 2014) and discourse relations (Lin et al., 2014). However, we found that only the following features are effective for recognizing insufficiently supported arguments:

Lexical: To capture lexical properties, we employ the 4,000 most frequent lowercased words as binary features analogous to SVM-bow.

Length: We use the number of tokens and the number of sentences as features since sufficiently supported arguments might exhibit more premises than insufficiently supported arguments (cf. Section 4.1.2).

²<https://www.languagetool.org/>

Syntax: For capturing syntactic properties, we extract binary production rules from the constituent parse trees of each sentence of the argument as described by Stab and Gurevych (2014).

Named Entities (ner): We assume that arguments with insufficient support refer to particular entities in order to justify more general claims (cf. example 1 in Section 1). Thus, we add the number of named entities appearing in the argument and the average occurrence of named entities per sentence to our feature set. We consider organizations, persons and locations separately. Thus the named entity features comprise six features in total, i.e. three binary and three numeric features.

5.3 Convolutional Neural Network (CNN)

Our second model is a convolutional neural network with max-over time pooling (Collobert et al., 2011). We use the implementation provided by Kim (2014). The selection of this model is motivated by the excellent performance that the model achieves in many different classification tasks like sentiment classification of question classification. We found in our experiments that instead of using several convolutional layers with different window sizes, a single convolutional layer with a window size of 2 and 250 feature maps performs best. For representing each word of an argument, we use word embeddings trained on the google news data set by Mikolov et al. (2013). In order to adapt these vectors to the identification of insufficient arguments, we use non-static word vectors as proposed by Kim (2014). We train the network with stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012), a dropout rate of .5 and a mini-batch size of 50. For finding the best model, we apply early stopping on the development sets.

5.4 Results

Table 2 shows the results of the model assessment on the test sets. The SVM-bow model with unigram features achieves .755 macro F1 score and .785 accuracy. It significantly outperforms the majority baseline by .357 macro F1 score which indicates that lexical features are informative for identifying insufficiently supported arguments. The support vector machine with manually created features significantly outperforms both the majority baseline and SVM-bow. It achieves .798 accuracy and .770 macro F1 score and thus outperforms the SVM-bow model by .015 macro F1

	<i>Accuracy</i>	<i>Macro F1</i>	<i>F1 Insufficient</i>	<i>Precision</i>	<i>Recall</i>
Human Upper Bound*	.911±.022	.887±.026	.940±.015	.863±.058	.808±.109
Baseline Majority	.662±.033	.398±.012	0	0	0
Baseline SVM-bow †	.785±.029	.755±.034	.661±.051	.709±.067	.624±.067
SVM †‡	.798±.028	.770±.032	.681±.047	.731±.060	.641±.061
CNN †‡	.843±.025	.827±.027	.770±.039	.762±.054	.784±.068

Table 2: Results of model assessment on the test sets and comparison to human upper bound († significant improvement over baseline majority; ‡ significant improvement over Baseline SVM-bow; *determined on a subset of 433 arguments).

score. We obtain the best performance by using the CNN model. It significantly outperforms all other models with respect to all evaluation scores and achieves .827 macro F1 score and an accuracy of .843. The results also show that the SVM model with manually created features achieves a considerably lower recall compared to precision. Thus, the model is less suitable for exhaustively finding all insufficiently supported arguments. In contrast, the CNN model is more balanced with respect to precision and recall and considerably outperforms the recall of the SVM model. Therefore, the CNN model outperforms the SVM model in finding insufficiently supported arguments in argumentative essay and performs better for recognizing arguments that are really in need of revision.

We determine the human upper bound by averaging the evaluation scores of all three annotator pairs on the 433 independently annotated arguments (cf. Section 4). Human annotators achieve an accuracy of .911. The CNN model yields only .068 less accuracy compared to the human upper bound and thus achieves 92.5% of human performance.

5.5 Feature Analysis

Although the CNN model outperforms the support vector machine with manual features, we analyzed the features for gaining a better understanding of insufficiently supported arguments and to investigate which linguistic properties are informative for recognizing arguments with insufficient support. Table 3 shows the macro F1 scores of the support vector machine using individual features and the results of feature ablation tests on the development sets.

The results show that lexical features are most effective for identifying insufficiently supported arguments. They achieve the best macro F1 score of .749 when used individually. Removing lexical features from the feature set also yields the highest

	<i>Macro F1</i>	<i>F1 Insuf.</i>	<i>F1 Suf.</i>
BS Majority	.396±.020	0	.793±.041
only lexical	.749±.048	.649±.070	.835±.040
only length	.397±.023	.002±.015	.792±.040
only syntax	.640±.063	.502±.101	.767±.047
only ner	.681±.059	.410±.114	.823±.039
all w/o lexical	.658±.059	.529±.093	.776±.045
all w/o length	.766±.049	.674±.068	.847±.040
all w/o syntax	.755±.049	.659±.070	.839±.040
all w/o ner	.760±.050	.666±.069	.843±.041
all features	.768±.049	.677±.068	.848±.040

Table 3: Results of the SVM using individual features and feature ablation tests on the dev sets.

decrease in macro F1 score compared to the other features. The second best features are named entities. Using only named entity features yields a macro F1 score of .681. Thus, we can confirm our assumption that named entities are informative features for assessing the sufficiency of arguments. Syntactic features are also effective for recognizing insufficiently supported arguments. They yield .640 macro F1 score when used individually. The results also show that the length of an argument is only marginally informative for assessing the sufficiency of arguments. Using the length features individually yields only a slight improvement of the macro F1 score over the majority baseline. However, removing the length from the entire feature set causes a slight decrease of .002 in the macro F1 score compared to the system which uses all features. We achieve the best results by combining all features.

For gaining further insights into the characteristics of insufficiently supported arguments, we ranked all unigrams using information gain. The top ten words are “example”, “my”, “was”, “instance”, “i”, “for”, “me”, “friend”, “he”, and “did”. This might be an indication that *examples* (signaled by the terms “example” and “instance”)

or *personal experiences* (signaled by terms such as “*me*”, “*my*”, “*friend*” or “*he*”) are not sufficient for developing strong arguments.

5.6 Error Analysis

In order to analyze the most frequent errors of the convolutional neural network, we manually investigated all arguments which are wrongly classified in each run of the repeated cross-validation experiment. In total, we found 41 sufficient arguments which are consistently misclassified as insufficient (false positives) and 28 insufficient arguments that are always misclassified as sufficient (false negatives).

Among the false positives, we observed that 35 arguments include examples as evidence which are signaled by terms like “*example*” or “*instance*”. Thus, the model tends to overemphasize the presence of particular lexical indicators. Most of these arguments either refer to an example in addition to other premises which are already sufficient to support the claim or include an example for specifying another premise. However, we also found several false negatives which include examples as evidence. Thus, the model does not solely rely on these lexical clues.

Among the 28 false negatives, we found 8 arguments that refer to multi-word named entities which are not captured by word embeddings. Another 5 false negatives support the claim by means of personal experience and 3 ones cite numbers, i.e. previous studies or empirical evidence.

6 Discussion

Although the convolutional neural network achieves promising results, the sufficiency criterion is only one of three criteria that a logically good argument needs to fulfill. Thus, our approach is not yet able to separate logically good from illogical arguments. In our experiments, we also analyzed arguments with respect to the relevance and acceptability criterion. In particular, we conducted several annotation studies with varying guidelines and two annotators on a set of 100 arguments. For annotating the relevance criterion, we presented the annotated structure of each argument to the annotators and asked them to assess the relevance of each premise for the claim individually. In order to evaluate the acceptability criterion, we asked the annotators to mark each premise as acceptable if it represents undisputed

common knowledge or a fact. However, we found that human annotators hardly agree on these criteria. We obtained low agreement scores of $\kappa = .435$ for the relevance criterion and $\kappa = .259$ for the acceptability criterion, which is not sufficient for creating a reliable corpus. In addition, we found that the violations of the relevance and acceptability criteria are less frequent than violations of the sufficiency criterion in argumentative essays. We observed that only 15% of the arguments include a premise that violates the relevance criterion and 14% of all premises violate the acceptability criterion. Although this imbalance explains the low agreement scores (Artstein and Poesio, 2008, p. 573), it also poses additional requirements for the size of the corpus and for computational models.

Although we didn’t obtain adequate agreement scores for the acceptability and relevance criteria, we implemented a system that identifies insufficiently supported arguments in argumentative essays with a reasonable accuracy. Given that sufficiency flaws are the most frequent quality defects in argumentative essays, our system represents an important milestone for realizing argumentative writing support systems.

7 Conclusion

We presented a novel approach for assessing the quality of natural language arguments. In particular, we focused on the sufficiency criterion that each logically good argument needs to fulfill. Previous approaches on argument quality are of limited use for argumentative writing support systems since they are not capable of recognizing particular weaknesses in argumentative texts. To overcome this limitation, we conducted an empirical study on the applicability of the sufficiency criterion to real arguments in argumentative essays. The inter-annotator agreement of $\alpha = .7673$ shows that human annotators substantially agree in this annotation task and confirms that humans can reliably separate sufficiently supported arguments from insufficiently supported arguments. We introduced a novel corpus annotated with the sufficiency criterion for studying logical mistakes in argumentation. This corpus is freely available for ensuring the reproducibility of our results and to encourage future research on argument quality. Furthermore, we presented the results of our experiments for automatically rec-

ognizing if an argument is sufficiently supported or not. We found that convolutional neural networks significantly outperform challenging baselines and manually created features with a macro F1 score of .827 and an accuracy of .843. Moreover, we showed that insufficiently supported arguments frequently exhibit particular lexical indicators. In addition, the feature analysis revealed that named entities and syntactic features are good indicators for separating sufficiently supported arguments from insufficiently supported arguments.

For future work, we plan to continue with our experiments with the relevance and acceptability criteria. In addition, we plan to integrate our method in writing environments for evaluating its effectiveness for supporting authors.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806 and by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus project AWS under grant No. 01|S12054. We thank our annotators Can Diehl and Radhika Gaonkar for their valuable contributions and the anonymous reviewers for their helpful comments.

References

- Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, Baltimore, MD, USA.
- Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.
- Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL '12, pages 208–212, Jeju Island, Korea.
- Amparo Elizabeth Cano-Basave and Yulan He. 2016. A study of the impact of persuasive argumentation in political debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1405–1413, San Diego, California.
- Lucas Carstens and Francesca Toni. 2015. Towards relation based argumentation mining. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 29–34, Denver, CO, USA.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- T. Edward Damer. 2009. *Attacking Faulty Reasoning: A Practical Guide to Fallacy-Free Reasoning*. Wadsworth Cengage Learning, 6th edition.
- Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40:1:9–56.
- Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: A Java-based framework for supervised learning experiments on textual data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. System Demonstrations*, ACL '14, pages 61–66, Baltimore, MD, USA.
- Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357.
- Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014*, pages 1–11, Dublin, Ireland.
- Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. 2015. On the role of discourse markers for discriminating claims and premises in argumentative discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pages 2236–2242, Lisbon, Portugal.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 363–370, Ann Arbor, Michigan.
- Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.
- Trudy Govier. 2010. *A Practical Study of Argument*. Wadsworth, Cengage Learning, 7th edition.
- Leo Groarke. 2015. Informal logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2015 edition.

- Ivan Habernal and Iryna Gurevych. 2016a. Argumentation mining in user-generated web discourse. *Computational Linguistics, arXiv preprint arXiv:1601.02403v4*, page (in press).
- Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, Berlin, Germany.
- Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.
- Ralph H. Johnson and Anthony J. Blair. 2006. *Logical Self-Defense*. International Debate Education Association.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP '14*, pages 1746–1751, Doha, Qatar.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1, ACL '03*, pages 423–430, Sapporo, Japan.
- Klaus Krippendorff. 1980. *Content Analysis: An Introduction to its Methodology*. Sage.
- Damjan Krstajic, Ljubomir J. Buturovic, David E. Leahy, and Simon Thomas. 2014. Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of Cheminformatics*, 6(10).
- Namhee Kwon, Liang Zhou, Eduard Hovy, and Stuart W. Shulman. 2007. Identifying and classifying subjective claims. In *Proceedings of the 8th Annual International Conference on Digital Government Research: Bridging Disciplines & Domains*, pages 76–81, Philadelphia, PA, USA.
- Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of the 25th International Conference on Computational Linguistics, COLING '14*, pages 1489–1500, Dublin, Ireland.
- Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.
- Christian M. Meyer, Margot Mieskes, Christian Stab, and Iryna Gurevych. 2014. Dkpro agreement: An open-source java library for measuring inter-rater agreement. In *Proceedings of the 25th International Conference on Computational Linguistics: System Demonstrations (COLING)*, pages 105–109, Dublin, Ireland.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.
- Raquel Mochales-Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law, ICAIL '09*, pages 98–107, Barcelona, Spain.
- Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law, ICAIL '07*, pages 225–230, Stanford, CA, USA.
- Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, MA, USA.
- Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP '15*, pages 938–948, Lisbon, Portugal.
- Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), ACL '15*, pages 543–552, Beijing, China.
- Chris Reed, Raquel Mochales-Palau, Glenn Rowe, and Marie-Francine Moens. 2008. Language resources for studying argument. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC '08*, pages 2613–2618, Marrakech, Morocco.
- Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP '15*, pages 440–450, Lisbon, Portugal.
- Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66, Denver, CO, USA.

- Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.
- Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, ACL '09, pages 226–234, Suntec, Singapore.
- Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78, Baltimore, MA, USA.
- Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 46–56, Doha, Qatar.
- Christian Stab and Iryna Gurevych. 2016. Parsing argumentation structures in persuasive essays. *arXiv preprint arXiv:1604.07370*.
- Christian Stab, Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Argumentation mining in persuasive essays and scientific articles from the discourse structure perspective. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 40–49, Bertinoro, Italy.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Avignon, France.
- Frans H. van Eemeren, Rob Grootendorst, and Francisca Snoeck Henkemans. 1996. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments*. Routledge, Taylor & Francis Group.
- Marilyn Walker, Jean Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 812–817, Istanbul, Turkey.
- Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 195–200, Berlin, Germany.
- Matthew D. Zeiler. 2012. Adadelata: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Distributed Document and Phrase Co-embeddings for Descriptive Clustering

Motoki Sato¹, Austin J. Brockmeier², Georgios Kontonatsios¹, Tingting Mu¹,
John Y. Goulermas², Jun'ichi Tsujii^{3,1} and Sophia Ananiadou¹

¹University of Manchester, National Centre for Text Mining (NaCTeM), Manchester, UK

²Department of Computer Science, University of Liverpool, Liverpool, UK

³Artificial Intelligence Research Center, AIST, Tokyo, Japan

sato.motoki.sa7@is.naist.jp, a.j.brockmeier@liverpool.ac.uk

sophia.ananiadou@manchester.ac.uk

Abstract

Descriptive document clustering aims to automatically discover groups of semantically related documents and to assign a meaningful label to characterise the content of each cluster. In this paper, we present a descriptive clustering approach that employs a distributed representation model, namely the paragraph vector model, to capture semantic similarities between documents and phrases. The proposed method uses a joint representation of phrases and documents (i.e., a co-embedding) to automatically select a descriptive phrase that best represents each document cluster. We evaluate our method by comparing its performance to an existing state-of-the-art descriptive clustering method that also uses co-embedding but relies on a bag-of-words representation. Results obtained on benchmark datasets demonstrate that the paragraph vector-based method obtains superior performance over the existing approach in both identifying clusters and assigning appropriate descriptive labels to them.

1 Introduction

Document clustering is a well-established technique whose goal is to automatically organise a collection of documents into a number of semantically coherent groups. Descriptive document clustering goes a step further, in that each identified document cluster is automatically assigned a human-readable label (either a word or phrase) that characterises the semantic content of the documents within the cluster. Descriptive clustering

methods have been shown to be useful in a variety of scenarios, including information retrieval (Bharambe and Kale, 2011), analysis of social networks (Zhao and Zhang, 2011), and large-scale exploration (Nassif and Hruschka, 2013) and visualisation of text collections (Kandel et al., 2012).

A number of previously proposed descriptive clustering techniques work by extending a standard document clustering approach. Documents are typically clustered based on a bag-of-words (BoW) representation (i.e., the occurrence counts of the words that appear in each document). Then, each cluster is labelled using the most commonly occurring word or phrase within the cluster (Weiss, 2006). In contrast to this approach, the recently proposed descriptive clustering approach (CEDL) (Mu et al., 2016) maps documents and candidate cluster labels into a common semantic vector space (i.e., co-embedding). The co-embedding space facilitates the straightforward assignment of descriptive labels to document clusters. The CEDL method has been shown to generate accurate cluster labels and achieved improved clustering performance when compared to standard descriptive clustering methods. Nonetheless, the co-embedding is based solely on a BoW representation of the documents and is thus limited in its ability to accurately represent the semantic similarity between documents.

In this paper, we investigate a specific case of descriptive clustering that selects a single multi-word phrase to characterise each cluster of documents (Li et al., 2008). Firstly, we assume descriptive phrases are to be selected from a candidate phrase set extracted from the corpus during preprocessing. The proposed method then follows the co-embedding descriptive clustering paradigm of the CEDL algorithm. However, in-

stead of using a BoW representation, we employ the paragraph vector (PV) (Le and Mikolov, 2014) model to learn a distributed vector representations of phrases and documents. These distributed representations move beyond unstructured BoW representations by considering the local contexts in which words and phrases appear within documents, which provides a more precise estimate of semantic similarity.

In particular, we present two extensions to the initial PV-based method that enable models that learn a common co-embedding space of documents and phrases. The first extension jointly learns co-embeddings of documents and phrases. The second extension constructs ‘pseudo-documents’ consisting of the lexical context surrounding each occurrence of a particular phrase. Each of these contexts are treated as separate document instance that are associated with a single embedded vector. In both cases, after clustering the document embedding vectors, each embedded phrase is a candidate cluster label. To select the most appropriate descriptive label amongst these candidates, we first rank the documents according to their proximity to each candidate label’s embedding vector and then select the phrase whose ranking maximises the average precision for a given cluster.

We compare the results obtained by our PV-based descriptive clustering method against two methods: spectral clustering (Shi and Malik, 2000), which only identifies clusters (but does not assign labels to them), and the previously introduced CEDL method (Mu et al., 2016), which carries out both clustering and labelling. Experimental results based on publicly available benchmark text collections demonstrate the effectiveness and superiority of our methods in both clustering performance and labelling quality.

2 Related Work

2.1 Descriptive Clustering

Descriptive clustering methods typically use an unsupervised approach to firstly group documents into flat or hierarchical clusters (Steinbach et al., 2000). Document clusters are then characterised using a set of informative and discriminative words (Zhu et al., 2006), phrases (Mu et al., 2016; Li et al., 2008) or sentences (Kim et al., 2015).

Early approaches to descriptive clustering

followed the description-comes-first (DCF) paradigm (Osiński et al., 2004; Weiss, 2006; Zhang, 2009). DCF-based methods work by firstly identifying a set of cluster labels, and subsequently forming document clusters by measuring the relevance of each document to a potential cluster label. DCF-based approaches have several shortcomings, which include poor clustering performance and low readability of cluster descriptors (Lee et al., 2008; Carpineto et al., 2009).

More recent developments in descriptive clustering have proposed alternative techniques, which approach the problems of improving clustering performance and descriptive label quality from various different angles. For instance, Scaiella et al. (2012) identifies Wikipedia concepts in documents and then computes relatedness between documents according to the linked structure of Wikipedia. Navigli and Crisafulli (2010) propose a method that takes into account synonymy and polysemy. Their method utilises the Google Web1T corpus to identify word senses based on word co-occurrences and computes the similarity between documents using the extracted sense information.

More recently, Mu et al. (2016) presented their co-embedding based descriptive clustering approach that learns a common co-embedding vector space of documents and candidate descriptive phrases. The co-embedded space simplifies the clustering and cluster labelling task into a more straightforward process of computing similarity between pairs of documents and between documents and candidate cluster labels.

2.2 Distributed Representation

Distributed representation techniques are becoming increasingly important in a number of supervised learning tasks, e.g., sentiment analysis (Dai et al., 2015), text classification (Dai et al., 2015; Ma et al., 2015) and named entity recognition (Turian et al., 2010). A number of models have been proposed to learn distributed word or phrase representations in order to predict word occurrences given a local context (Mnih and Hinton, 2009; Mikolov et al., 2013b; Mikolov et al., 2013a; Pennington et al., 2014). Subsequently, the PV model was proposed to learn representations of both words and documents (Le and Mikolov, 2014; Dai et al., 2015). The PV model has been

shown to be capable of learning a semantically richer representation of documents compared to unstructured BoW models. To our knowledge, our work constitutes the first attempt to use distributed representation models to co-embed documents and phrases for unsupervised descriptive clustering.

3 Proposed Descriptive Clustering Method

As outlined above, the descriptive clustering task (i.e., grouping documents according to semantic relatedness and characterising the cluster content using a representative descriptive phrase) relies heavily on learning a representation of documents and phrases that can accurately capture relevant semantic information. A particularly effective strategy for descriptive clustering is to jointly map documents and descriptive phrases together into a common embedding space (Mu et al., 2016). The clustering of documents and selection of descriptive phrases for each cluster is then carried out by calculating the cosine similarities between documents (to form clusters), and between documents and descriptive phrases (to determine descriptive labels) in the learned space. Instead of relying on the commonly used BoW model, we propose a novel descriptive clustering approach. Our method uses similarities computed from distributed joint embeddings of documents and phrases, which are learned by considering both the global context provided by the document and the local context of the descriptive phrases. We propose two different strategies to learn these embeddings, as described below.

3.1 Joint Learning of Document and Phrase Embeddings

The first strategy jointly learns the distributed representations for documents and phrases by representing phrases, words, and documents as vectors that are used both to predict the occurrence of words in given documents (reflecting global document content information), and to predict the co-occurrences of words and phrases within a sliding window, to reflect the local context information.

We extend the PV model described in Dai et al. (2015) to simultaneously generate word, phrase and document embeddings. The objective function is to maximise the log probability of words and phrases conditioned on either their global or

local context:

$$\begin{aligned} & \sum_{t \in \mathcal{T}_P} \log p(p_t | d_t) + \frac{1}{|\mathcal{C}_t|} \sum_{c \in \mathcal{C}_t} \log p(p_t | c) \quad (1) \\ & + \sum_{s \in \mathcal{T}_W} \log p(w_s | d_s) + \frac{1}{|\mathcal{C}_s|} \sum_{c \in \mathcal{C}_s} \log p(w_s | c) \end{aligned}$$

where \mathcal{T}_P is the set of training phrase instances; $p_t \in \mathcal{P}$ is the t -th phrase instance; d_t denotes the document corresponding to the t -th training instance; c denotes a member of the local context $\mathcal{C}_t = [q_{t-L}, \dots, q_{t-1}, q_{t+1}, \dots, q_{t+L}]$, which occurs within a window size of L of the training instance ($|\mathcal{C}_t| = 2L$) and consists of both words and phrases $q_t \in \mathcal{P} \cup \mathcal{W}$; likewise, \mathcal{T}_W is the set of training word instances; $w_s \in \mathcal{W}$ is the s -th target word instance; d_s denotes the document corresponding to the s -th training instance; and \mathcal{C}_s is its local context with $|\mathcal{C}_s| = 2L$. To summarise, the probability terms $p(p_t | d_t)$ and $p(w_s | d_s)$ model the document content information from a global level, while $p(p_t | c)$ and $p(w_s | c)$ model the local context. There are $2L + 1$ conditional probabilities estimated for each training instance.

The probability of a given lexical unit $q_t \in \mathcal{P} \cup \mathcal{W}$ (either a word or a phrase) is modelled using the vector embeddings of the $|\mathcal{P}| + |\mathcal{W}|$ words and phrases and the softmax function as follows:

$$p(q_t | d_t) = \frac{\exp(\mathbf{u}_{q_t}^\top \mathbf{z}_{d_t})}{\sum_{q \in \mathcal{P} \cup \mathcal{W}} \exp(\mathbf{u}_q^\top \mathbf{z}_{d_t})} \quad (2)$$

$$p(q_t | c) = \frac{\exp(\mathbf{u}_{q_t}^\top \mathbf{z}_c)}{\sum_{q \in \mathcal{P} \cup \mathcal{W}} \exp(\mathbf{u}_q^\top \mathbf{z}_c)} \quad (3)$$

where \mathbf{u}_{q_t} is a weight vector specific to the target word or phrase, \mathbf{z}_{d_t} is the embedding vector of the document corresponding to instance t , and \mathbf{z}_c is the embedding vector of a word or phrase in the context of q_t . Since the document, phrase and word vectors all use the same weight vector \mathbf{u}_{q_t} to predict the target phrase, they are necessarily in the same vector space.

3.2 Phrase Embeddings via Local Context Pseudo-Documents

The previous model considers learning an embedding as a multi-objective problem by trying to predict phrases and words based on the global and local context. Besides indexing, Equation (1) treats words and descriptive phrases interchangeably. An alternative approach is to treat phrases

as ‘pseudo-documents’ by using the sets of words appearing in the local context of each phrase occurrence. Specifically, training instances for a phrase’s embedding vector are constructed by extracting the local context around each occurrence of a phrase in the document collection. Using the augmented training set, consisting of both the original documents and the additional pseudo-documents, we can then employ any existing PV model (Le and Mikolov, 2014; Dai et al., 2015) to learn the document-phrase co-embeddings.

However, due to the significant differences in the sizes and numbers of documents and pseudo-documents, there is a danger that the addition of the pseudo-documents can have a detrimental effect on the performance of the model. Thus, we adopt a two-stage training procedure. Firstly, an embedding model is trained using only the documents. Then, we fix the weights of the model and optimise the phrase embeddings by providing the pseudo-documents as the input to the model.

We have integrated the above-mentioned process with two PV approaches, namely the distributed memory model (PV-DM) Le and Mikolov (2014), and the extension of the distributed BoW model (PV-DBOW) in Dai et al. (2015).

In the PV-DM model, the probability that a target word will appear in a given lexical context is conditioned on the surrounding co-occurring words and also the document:

$$\sum_{t \in \mathcal{T}_W} \log p(w_t | \mathcal{C}_t, d_t), \quad (4)$$

where w_t is the target word for instance t , \mathcal{T}_W is the set of training word instances, $\mathcal{C}_t = [w_{t-L}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+L}]$ are context words that occur within a window size of L words around w_t , and d_t denotes the document corresponding to the t -th training instance. The probability is modelled using a softmax function.

For phrase p , the objective is to maximise the sum of the log probabilities $\sum_{t \in \mathcal{T}_p} \log p(w_t | \mathcal{C}_t, p)$ where $w_t \in \mathcal{T}_p$ are the word instances that appear in local context around the phrase, i.e., \mathcal{T}_p is the set of word instances across all pseudo-documents, and \mathcal{C}_t is the set of words that occur around the t -th word instance which also occur within the pseudo-documents for the phrase. Explicitly, the optimal embedding vector for the phrase is determined by

solving the following optimisation problem:

$$\max_{\mathbf{z}_p} \sum_{t \in \mathcal{T}_p} \log \frac{\exp(\mathbf{u}_{w_t}^\top \mathbf{x}_t + \mathbf{v}_{w_t}^\top \mathbf{z}_p)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{x}_t + \mathbf{v}_w^\top \mathbf{z}_p)} \quad (5)$$

$$\mathbf{x}_t = [\mathbf{x}_{w_{t-L}}^\top, \dots, \mathbf{x}_{w_{t-1}}^\top, \mathbf{x}_{w_{t+1}}^\top, \dots, \mathbf{x}_{w_{t+L}}^\top]^\top$$

where \mathbf{x}_t is the concatenation of all word vectors in the context of word w and $\{\mathbf{u}_w\}_w$ and $\{\mathbf{v}_w\}_w$ for w . To find an approximate solution, the parameters of the embedding vector are randomly initialised and optimised using stochastic gradient descent; the gradient is calculated via backpropagation (Rumelhart et al., 1986).

The PV-DBOW model simplifies the PV-DM model by ignoring the local context of words in the log probability function. The probability that a target word will appear in a given lexical context is conditioned solely by the document. Dai et al. (2015) introduced a modified version of the PV-DBOW model that treats words and documents as interchangeable inputs to the neural network. This enables the model to jointly learn word and document embeddings in the same space; we denote the model as PV-DBOW-W. Essentially, the objective of the PV-DBOW-W model is a combination of both the skip-gram model (Mikolov et al., 2013b) that generates word embeddings and the PV-DBOW method which is used for learning document embeddings:

$$\sum_{t \in \mathcal{T}_W} \log p(w_t | d_t) + \frac{1}{|\mathcal{C}_t|} \sum_{c \in \mathcal{C}_t} \log p(w_t | c). \quad (6)$$

To optimise the embedding of a specific phrase, denoted p , the existing word embeddings remain fixed, and the objective function is simplified as $\sum_{t \in \mathcal{T}_p} \log p(w_t | p)$ where $w_t \in \mathcal{T}_p$ are word instances that appear in the local contexts around the phrase. The optimal embedding vector for this phrase is determined by solving the following optimisation problem:

$$\max_{\mathbf{z}_p} \sum_{t \in \mathcal{T}_p} \log \frac{\exp(\mathbf{u}_{w_t}^\top \mathbf{z}_p)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{z}_p)} \quad (7)$$

where the weight vectors $\{\mathbf{u}_w\}_w$ are fixed. As in the previous model, the parameters of the embedding vector are randomly initialised and optimised using stochastic gradient descent.

3.3 Descriptive Phrase Selection

Given co-embeddings of documents and phrases, any clustering algorithm can be applied. We use k-means, with the cosine similarity-based distance metric, to cluster the documents. Given the set of documents within each identified cluster $\mathcal{G}_1, \dots, \mathcal{G}_K$, the document embedding vectors $\{\mathbf{z}_d\}_{d=1}^N$ and the descriptive phrase embedding vectors $\{\mathbf{z}_p\}_{p=1}^P$, we then select a descriptive phrase that best represents the documents assigned to a cluster.

A baseline approach for descriptive phrase selection is to select the phrase whose embedding vector is nearest to the cluster centroid; however, proximity to the cluster centroid is not always a good indicator of cluster membership, as it ignores the location of documents belonging to other clusters. An ideal phrase vector should lie closer to documents within the cluster than documents outside of the cluster. Accordingly, we rank documents based on their proximity to a candidate phrase and calculate the average precision of this ranking (where documents belonging to the given cluster are the true positives).

For cluster \mathcal{G} , we define the cluster membership indicator for each document as:

$$y_d = \begin{cases} 1 & d \in \mathcal{G} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

For a given phrase p , let $\pi_p(1)$ be the index of the nearest document to the phrase, and $\pi_p(i)$ be the index of the i -th nearest neighbour. The precision after the k -nearest documents are retrieved is $P_{\pi_p}(k) = \frac{1}{k} \sum_{i=1}^k y_{\pi_p(i)}$. The phrase which maximises the average precision \bar{P}_p is selected as the cluster descriptor

$$p^* = \arg \max_p \left\{ \bar{P}_p = \frac{1}{|\mathcal{G}|} \sum_{k \in |\mathcal{G}|} P_{\pi_p}(k) \right\}, \quad (9)$$

where $|\mathcal{G}|$ is the number of documents in the cluster.

4 Results

We evaluate the proposed PV-based descriptive clustering methods in terms of cluster quality and descriptive phrase selection. Additionally, we show a visualisation of the co-embedding space in the supplementary material.

4.1 Datasets

We use two well-known, publicly available datasets: ‘Reuters-21578 Text Categorization Test Collection’ from the Reuters newswire (Lewis, 1997), and the ‘20 Newsgroups’ email dataset¹. We pre-process the 20 Newsgroups corpus to remove email header information while for both datasets we extract candidate phrases using Termine (Frantzi et al., 2000), an automatic term extraction tool.

For the Reuters corpus, we use the complete document collection for training the PV models. For evaluation, we use both the training and testing sets of the modApte split, and select the 10 categories with the largest number of documents. Moreover, we remove documents that belong to multiple categories, this process results in an evaluation set of 8,009 documents. For the 20 Newsgroups dataset, we use the complete set of 18,846 documents for training the PV models. We remove words and phrases that only appear in a single document and then remove any empty documents. This process results in an evaluation set of 18,813 documents with 20 categories, organised into 4 higher level parent categories. Table 1 summarises various characteristics of the employed datasets, including: a) number of documents, b) number of candidate phrases and c) category labels.

Table 1: Categories included in the evaluation subsets. ‘R10’ corresponds to the 10 largest categories after removing documents with multiple categories; the number of documents is in parentheses. All 20-Newsgroups categories have between 628 and 997 documents.

Reuters - 8,009 docs - 9,984 words - 11,732 phrases	
R10	earn(3923), acq(2292), crude(374), trade(327), money-fx(293), interest(271), money-supply(151), ship(144), sugar(122), coffee(112)
20 News - 18,813 docs - 43,285 words - 36,041 phrases	
sci	crypt, electronics, med, space
comp	os.ms-windows.misc, sys.ibm.pc.hardware, graphics, windows.x, sys.mac.hardware
rec	autos, motorcycles, sport.baseball, sport.hockey
mix	comp.os.ms-windows.misc, rec.autos, rec.sport.baseball, sci.med, sci.space
all	*

4.2 Paragraph Vector Models

In this section, we provide implementation details for the three PV models (PV-DBOW-WP,

¹<http://qwone.com/~jason/20Newsgroups/>

PV-DBOW-W, and PV-DM), introduce a fourth model (PV-CAT) and explain the different settings that we use throughout the experiments. The PV-DBOW-WP model is used to jointly train phrase, word and document co-embeddings. For the PV-DBOW-W and PV-DM models, we use the two-stage training approach, in which the document embeddings and softmax weights are trained first, and then the phrase co-embeddings are trained using pseudo-documents. A window size of 10 words around the target phrase is used as the local context to create the pseudo-documents.

Each PV model has a number of parameters, including the dimension of the embedded spaces and the size of the context window. We set all embedding dimensions to 100. For the PV-DBOW-W and PV-DBOW-WP model, we use a context window of 10 words/phrases while for the PV-DM model a window size of 2 words (we tuned the size of the context window by applying the two PV models to a small development set of the Reuters corpus). This disparity in window size is not surprising since the PV-DM model considers the order of words within the local context and uses different parameters for the vectors at each location in the context window, Equation (5), whereas an increased window size does not add additional parameters to the PD-DBOW model.

We create an additional model, namely PV-CAT, by concatenating the vector representations induced by the PV-DBOW-W and the PV-DM models. This is performed after training the document and the phrase vectors. Intuitively, the concatenation of the PV-DBOW-W and PV-DM feature vectors can provide complimentary information given that the two models are trained using a different size of context window (i.e., 10 and 2 words, respectively).

Given that the size of the vocabulary is very large, computing the softmax function during stochastic gradient descent is computationally expensive. For faster training, different optimisation algorithms can be used to approximate the log probability function. We use a combination of negative sampling and hierarchical softmax via backpropagation (Mnih and Hinton, 2009; Mikolov et al., 2013b). Specifically, we use negative sampling and then further optimise the embeddings using hierarchical softmax. Although, these are different optimisation approaches, both methods can be applied in this ad-hoc manner.

Moreover, we follow the process described in Le and Mikolov (2014) to tune the learning rate. For this, we set the initial learning rate to 0.025 and decrease it linearly during 10 training epochs such that the learning rate is 0.001 during the last training epoch.

4.3 Baseline Methods

As our first baseline, we perform spectral clustering based on the affinity matrix produced according to the cosine similarity between the standard term-frequency inverse document (tf-idf) representation of the documents. We used the normalised cut (NC) spectral clustering algorithm proposed by Shi and Malik (2000).

We also compare our proposed method to the CEDL algorithm (Mu et al., 2016), which uses a measure of second-order similarity between phrases and documents, based on their co-occurrences at the document level, to obtain a spectral co-embedding. We use the same parameters suggested in the original publication, but carried out minor changes to the algorithm to allow the method to be scaled up to larger datasets. To compare clustering performance, we also run the CEDL algorithm without the phrase co-embeddings.

4.4 Evaluation of Cluster Quality

In this experiment, we evaluate the clustering performance of the methods by comparing automatically generated document clusters against the gold standard categories. For all methods, we use k-means clustering with cosine similarity as the distance metric. Following previous approaches (Xie and Xing, 2013), we set the number of clusters equal to the number of gold standard categories. As evaluation metrics, we use the macro-averaged F1 score², and normalised mutual information³.

Table 2 compares the clustering performance achieved by four PV models (PV-DBOW-WP, PV-DBOW-W, PV-DM, and PV-CAT) against the performance of the baselines (i.e., the two versions of the CEDL algorithm and spectral clustering via normalised cut).

²For each category, the maximum F1 score across the clusters is used.

³Normalised mutual information $\frac{MI(G,C)}{\max\{H(G),H(C)\}}$ is defined as the mutual information between the automatically generated clusters and gold standard categories $MI(G,C)$ divided by the maximum of the entropy of the clusters $H(G)$ or the categories $H(C)$.

Table 2: Clustering performance assessed by correspondence measures to gold standard categories. *NC*: spectral clustering via normalised cut, *CE1*: CEDL, *CE2*: CEDL without phrase co-embeddings, *PV1*: PV-DBOW-WP, *PV2*: PV-DBOW-W, *PV3*: PV-DM, *PV4*: PV-CAT.

	<i>NC</i>	<i>CE1</i>	<i>CE2</i>	<i>PV1</i>	<i>PV2</i>	<i>PV3</i>	<i>PV4</i>
F1 (macro-averaged; higher is better)							
R10	0.60	0.54	0.56	0.54	0.66	0.48	0.66
sci	0.89	0.89	0.88	0.91	0.91	0.90	0.92
comp	0.48	0.46	0.55	0.67	0.66	0.63	0.65
rec	0.87	0.86	0.90	0.92	0.92	0.88	0.92
mix	0.93	0.92	0.93	0.95	0.94	0.93	0.95
all	0.56	—	—	0.69	0.69	0.66	0.69
all [†]	0.52	0.48	0.55	0.67	0.67	0.64	0.67
Normalised mutual information (higher is better)							
R10	0.42	0.41	0.42	0.47	0.50	0.40	0.51
sci	0.68	0.68	0.66	0.72	0.71	0.70	0.74
comp	0.25	0.22	0.30	0.40	0.39	0.36	0.39
rec	0.69	0.67	0.74	0.78	0.78	0.70	0.78
mix	0.80	0.77	0.79	0.84	0.82	0.80	0.83
all	0.51	—	—	0.62	0.62	0.60	0.63
all [†]	0.50	0.44	0.52	0.63	0.62	0.60	0.64

[†] Average of 10 random subsets using 10% of each category’s documents.

The PV-DBOW-W and PV-CAT methods yield the best clustering performance on the Reuters dataset. Performance gains over the three baseline methods range between 6% – 12% (F1 score) and 8% – 9% (normalised mutual information). On the 20 Newsgroups dataset, the PV-DBOW-WP and PV-CAT models outperformed the baseline methods⁴ by approximately 2% – 21% (F1 score) and 3% – 18% (normalised mutual information). Moreover, we note that the performance achieved by the PV-CAT model exceeds the best results reported in Xie and Xing (2013) (normalised mutual information of 0.6159 using a multi-grain clustering topic model).

Finally, we note that co-embedding is not designed as a mechanism for improving cluster quality. For CEDL, the co-embedding of phrases reduced the clustering performances in most datasets. This reduction in performance is equally observed for the paragraph vector models when applied to the Reuters dataset: the jointly trained co-embedding model (i.e., PV-DBOW-WP) achieved a lower performance than the two-stage approach PV-DBOW-W (F1 score of 0.56 and 0.66, respectively).

⁴Our implementation of the CEDL algorithm did not scale up to the entire dataset (‘all’), but average results on random subsets were consistent, as shown in the table.

4.5 Evaluation of Cluster Labelling

In this section, we evaluate the cluster labels (i.e., multi-word phrases) selected by the proposed PV-based descriptive clustering methods. As a baseline approach, we use the CEDL algorithm that produces a co-embedded space of documents and phrases⁵.

For each document cluster, we apply the phrase selection criterion, Equation (9), to identify the phrase that best describes the underlying cluster. Then for each gold standard category, the cluster having the highest proportion of documents belonging to the category is determined. This process means some clusters are assigned to multiple categories while other clusters are left unassigned. For each assigned cluster, we rank all documents according to their similarity to the automatically selected phrase (in the co-embedding space), where documents within the cluster have precedence over documents outside the cluster. We evaluate the quality of the cluster label by computing the average precision of this ranking in recalling the gold standard category. The average precision is maximised when documents closest to the selected phrase belong to the gold standard category. Table 3 shows the selected cluster descriptors aligned to the gold standard categories, the average precision and mean average precision scores achieved by the CEDL method and PV-based descriptive clustering models when applied to the Reuters and the 20 Newsgroups dataset.

The Reuters dataset presents a challenging case for descriptive clustering methods given that the distribution of gold standard categories is highly skewed, i.e., the majority categories (e.g., ‘earn’ and ‘acq’) correspond to more than one clusters while the remaining clusters cover multiple smaller categories. Nonetheless, we observe that the automatically selected cluster descriptors are related to the corresponding gold standard categories (e.g., ‘import coffee’ and ‘oil export’ for gold standard category ‘ship’). In practice, the skewed distribution of gold standard categories can be addressed by using a larger number of clusters in k-means, or by using a cluster algorithm more amenable to heterogeneously sized clusters.

The 20 Newsgroups dataset shows a more balanced distribution of categories than the Reuters

⁵The spectral clustering via normalised cut and the CEDL algorithm without document and phrase co-embeddings are document clustering methods but not descriptive clustering models and thus they are excluded from this experiment.

Table 3: Cluster descriptions and average precision (as percentages) achieved by descriptive clustering methods. *CE1*: CEDL, *PV1*: PV-DBOW-WP, *PV2*: PV-DBOW-W, *PV3*: PV-DM, *PV4*: PV-CAT. The average precision metric depends on not only the phrase but also the location of documents relative to the selected phrase; consequently, the average precision of a phrase may vary among the embeddings.

Category	<i>CE1</i>	<i>PV1</i>	<i>PV2</i>	<i>PV3</i>	<i>PV4</i>					
earn	memotec datum	85	payable april	76	mln oper rev	89	mth jan	77	mln oper rev	88
acq	undisclosed sum	80	tender offer	58	undisclosed sum	73	tender offer	70	definitive agreement	70
crude	opec oil	92	venezuelan crude oil	47	oil production	88	oil export	79	oil production	91
trade	european community	49	trade relation	48	trade problem	75	representative clayton	77	trade problem	79
money-fx	bank rate	18	commercial lending rate	13	trade problem	17	deposit rate	22	trade problem	16
interest	bank rate	68	commercial lending rate	36	week t	49	deposit rate	53	week t	45
ship	european community	8	import coffee	7	raw sugar	15	oil export	11	costa rica	14
sugar	european community	67	import coffee	17	raw sugar	79	representative clayton	8	costa rica	34
money-supply	bank rate	9	commercial lending rate	15	week t	33	deposit rate	21	week t	37
coffee	european community	8	import coffee	43	raw sugar	25	representative clayton	8	costa rica	68
Unused clusters:	furman selz report		improve earning		share takeover offer		definitive agreement		share takeover offer	
	loss nil		undisclosed term		high earning		april record		revenue growth	
	bid null		group turnover		tax profit		exclude loss		april record	
	record today		current qtr		april record		mln net		mln dividend	
	present intention						net shr profit			
Mean average precision		48		36		54		42		54
sci.crypt	clipper key	93	key registration	94	back door	95	encryption method	95	back door	96
sci.electronics	transistor circuit	87	power amp	90	voltage divider	89	radio shack	86	circuit board	90
sci.med	other doctor	93	tech people	95	other treatment	94	other symptom	94	other treatment	95
sci.space	earth orbit	94	deep space	95	first spacecraft	95	low earth orbit	94	low earth orbit	96
comp.os.ms-windows.misc	trident 8900c	24	enhance mode	62	ms speaker sound driver	60	window version	55	dos app	59
comp.graphics	image file	44	art scene	70	art scene	74	swim chip	68	facet based modeller	70
comp.sys.ibm.pc.hardware	scsi hard drive	47	ide drive	60	tape drive	51	cmos setup	55	tape drive	52
comp.windows.x	application code	56	other widget	85	return value	77	widget name	79	return value	78
comp.sys.mac.hardware	apple price	59	mac lc	69	apple price	63	extra box	47	apple price	63
rec.autos	auto car	92	luxury sedan	94	same car	92	new car	89	sport car	94
rec.motorcycles	other bike	95	cruiser rider	95	same site	94	dod ama icoa nia	89	waterski bike	95
rec.sport.baseball	padded bat	88	leadoff hitter	94	playoff team	95	total baseball	91	playoff team	95
rec.sport.hockey	hockey playoff	92	cup final	94	nhl results	96	cup final	90	cup final	95
comp.os.ms-windows.misc	dos window font	95	auto show	97	dos window	97	dos window	97	dos window	98
rec.autos	bmw car	96	other car	97	same car	96	new car	97	other car	97
rec.sport.baseball	baseball fan	98	worst team	99	more game	98	last season	98	baseball season	99
sci.med	other doctor	94	other medical problem	96	many patient	94	other symptom	93	many patient	95
sci.space	earth orbit	94	japanese space agency	95	solar power	94	lunar surface	94	solar power	96
Mean average precision		80		88		86		84		87

corpus, and we note that all descriptive clustering methods were able to identify meaningful cluster descriptors that have a clear correspondence to the gold standard categories (e.g., ‘window version’ and ‘dos app’ for the category ‘comp.os.ms-windows.misc’).

With regard to the mean average precision, we observe that the PV-DBOW-W and PV-CAT models obtained the best performance. Moreover, the PV-CAT model achieved statistically significant improvements over the CEDL baseline in terms of the average precision across the 28 categories while no significant⁶ improvement was observed for the remaining three PV-based models.

The results that we obtained demonstrate that the PV-based co-embedding space can effectively capture semantic similarities between documents and phrases. An illustrative example of this is shown in Table 4. In this example, we selected two documents that neighbour the phrase “user interface” in the PV-CAT co-embedded feature space

⁶For significance testing, we used a paired sign-test, with a significance threshold of 0.05 and Bonferroni multiple test correction for the 4 tests; the uncorrected p-value for the PV-CAT model is 0.0009.

for the “20 Newsgroups” dataset. It can be noted that although neither of the two documents explicitly contain the input phrase, the first discusses a semantically similar topic, and the second uses the acronym GUI, i.e., graphical user interface.

As another example, we generate a two-dimensional visualisation of the document-phrase co-embeddings using t-SNE (van der Maaten and Hinton, 2008) that demonstrates how co-embedded phrases can be used as ‘landmarks’ for exploring a corpus. For this example, we use the ‘sci’ categories from the 20 Newsgroup corpus and select the 200 most frequent phrases in this subset. As input to t-SNE, we use the chordal distance defined by the cosine similarity in the co-embedding space and set the perplexity level to 40. Figure 1 in the supplementary material shows the visualisation with the cluster boundaries, location of the documents and co-embedded phrases.

5 Conclusion

Descriptive document clustering helps information retrieval tasks by automatically organising document collections into semantically coherent groups and assigning descriptive labels to each

Table 4: Two documents whose vector embeddings were the 5th and 6th nearest neighbours (according to the cosine of the angle of the corresponding vectors) to the phrase “user interface” in the PV-CAT based co-embedded space.

train/comp.windows.x_67337
<i>Does anyone know the difference between MOOLIT and OLIT? Does Sun support MOOLIT? Is MOOLIT available on Sparcstations? MoOLIT (Motif/Open Look Intrinsic Toolkit allows developers to build applications that can switch between Motif and Open Look at run-time, while OLIT only gives you Open Look.</i>
<i>Internet: chunhong@vnet.ibm.com</i>
test/comp.windows.x_68238
<i>Hi there, I'm looking for tools that can make X programming easy. I would like to have a tool that will enable to create X motif GUI Interactively. Currently I'm Working on a SGI with forms. A package that enables to create GUI with no coding at all (but the callbacks). Any help will be appreciated. Thanks Gabi.</i>

group. In this paper, we have presented a descriptive clustering method that uses paragraph vector models to support accurate clustering of documents and selection of meaningful and precise cluster descriptors. Our PV-based approach maps phrases and documents to a common feature space to enable the straightforward assignment of descriptive phrases to clusters. We have compared our approach to another state-of-the-art algorithm employing a co-embedding based on bag-of-word representations. The PV-based descriptive clustering method achieved superior clustering performance on both the Reuters and the 20 Newsgroups datasets. An evaluation of the selected cluster descriptors showed that our method selects informative phrases that accurately characterise the content of each cluster.

Acknowledgments

This research was partially funded by the Medical Research Council grant MR/L01078X/1 “Supporting Evidence-based Public Health Interventions using Text Mining”.

References

- Ujwala Bharambe and Archana Kale. 2011. Landscape of web search results clustering algorithms. In *Advances in Computing, Communication and Control*, pages 95–107. Springer.
- Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009. A survey of

web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17.

- Andrew M. Dai, Christopher Olah, and Quoc V. Le. 2015. Document embedding with paragraph vectors. *CoRR*, abs/1507.07998.
- Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms: The c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130.
- Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 547–554.
- Sun Kim, Lana Yeganova, and John W. Wilbur. 2015. Summarizing topical contents from PubMed documents using a thematic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 805–810. Association for Computational Linguistics.
- Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.
- Yeha Lee, Seung-Hoon Na, and Jong-Hyeok Lee. 2008. Search result clustering using label language model. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.
- David D. Lewis. 1997. Reuters-21578 text categorization test collection, distribution 1.0. Available at <https://archive.ics.uci.edu/ml>.
- Yanjun Li, Soon M. Chung, and John D. Holt. 2008. Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64(1):381–404.
- Chenglong Ma, Weiqun Xu, Peijia Li, and Yonghong Yan, 2015. *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, chapter Distributional Representations of Words for Short Text Classification, pages 33–38. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *Workshop at International Conference on Learning Representations (ICLR)*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

- Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088.
- Tingting Mu, John Y. Goulermas, Ioannis Korkontzelos, and Sophia Ananiadou. 2016. Descriptive document clustering via discriminant learning in a co-embedded space of multilevel similarities. *Journal of the Association for Information Science and Technology*, 67(1):106–133.
- C. Filipe Nassif and Eduardo Raul Hruschka. 2013. Document clustering for forensic analysis: An approach for improving computer inspection. *Information Forensics and Security, IEEE Transactions on*, 8(1):46–54.
- Roberto Navigli and Giuseppe Crisafulli. 2010. Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 116–126. Association for Computational Linguistics.
- Stanisław Osiński, Jerzy Stefanowski, and Dawid Weiss. 2004. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Information Processing and Web Mining*, pages 359–368. Springer.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.
- Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. 2012. Topical clustering of search results. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 223–232.
- Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905.
- Michael Steinbach, George Karypis, and Vipin Kumar. 2000. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, pages 525–526.
- Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.
- Dawid Weiss. 2006. *Descriptive Clustering as a Method for Exploring Text Collections*. Ph.D. thesis, Poznan University of Technology, Poznań, Poland.
- Pengtao Xie and Eric P. Xing. 2013. Integrating document clustering and topic modeling. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*.
- Chengzhi Zhang. 2009. Document clustering description based on combination strategy. In *Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, pages 1084–1088. IEEE.
- Peixin Zhao and Cun-Quan Zhang. 2011. A new clustering method and its application in social networks. *Pattern Recognition Letters*, 32(15):2109–2118.
- Ye-Hang Zhu, Guan-Zhong Dai, Benjamin C.M. Fung, and De-Jun Mu. 2006. Document clustering method based on frequent co-occurring words. In *Proceedings of the 20th Pacific Asia Conference on Language, Informatics, and Computation (PACLIC)*, pages 442–445.

A Supplementary Material

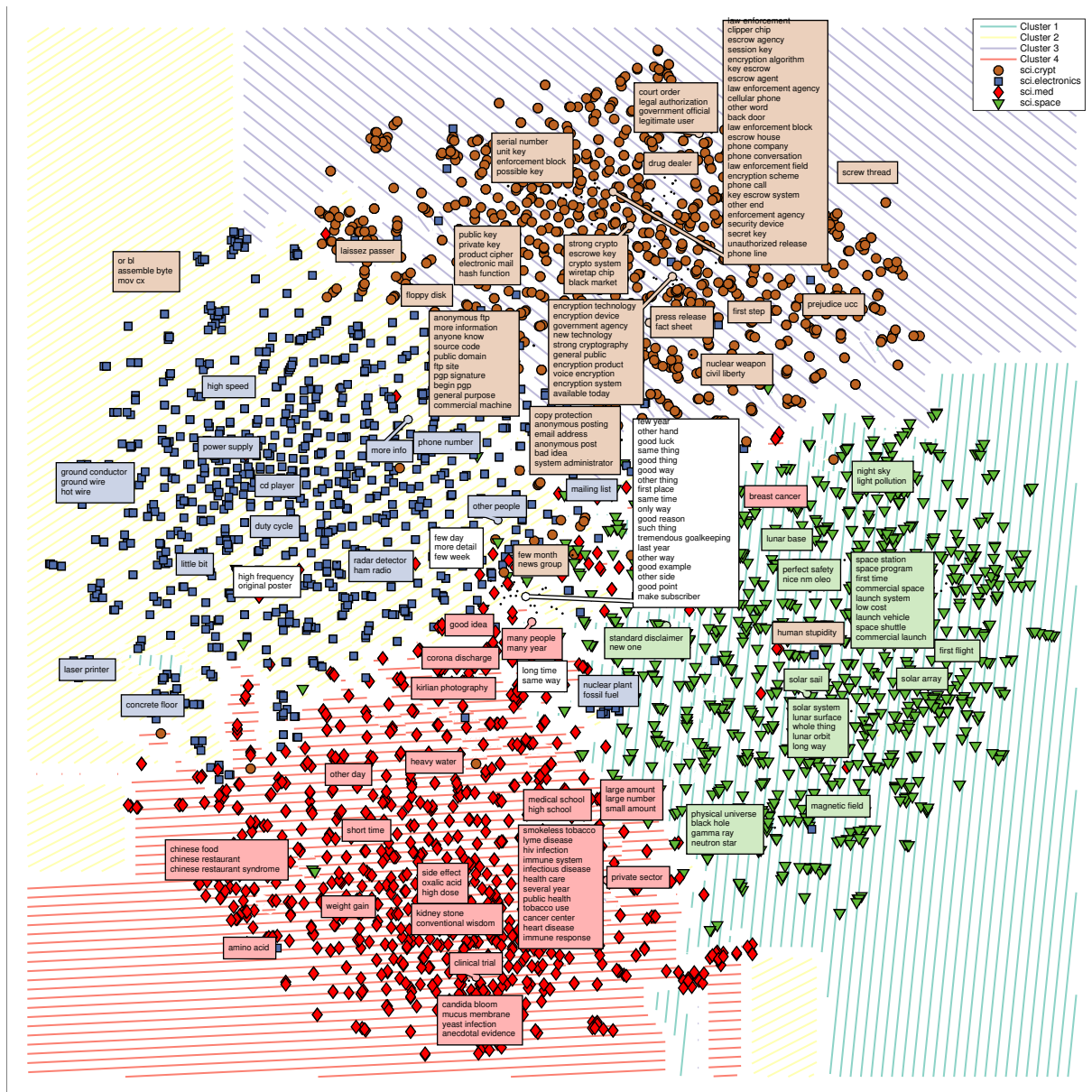


Figure 1: t-SNE visualisation of the PV-CAT co-embedding of the 20 Newsgroups ‘sci’ categories. Documents are represented by markers corresponding to the gold standard categories. Text boxes show the 200 most frequent phrases with nearby co-embedded phrases aggregated together. To show the correspondence between the categories and the phrase embedding, the text boxes are coloured based on the majority category of the documents nearest to each phrase. Sets of phrases with no majority category are left white. Hatch lines in the background denote the boundaries of each cluster, where the hatch angle (and colour) is based on the cluster. In the embedding space, each cluster is a convex set, but the t-SNE algorithm preserves local neighbourhoods and may fragment the clusters. (Best viewed with digital magnification.)

SMARTies: Sentiment Models for Arabic Target entities

Noura Farra and Kathleen McKeown

Columbia University

Department of Computer Science

{noura, kathy}@cs.columbia.edu

Abstract

We consider entity-level sentiment analysis in Arabic, a morphologically rich language with increasing resources. We present a system that is applied to complex posts written in response to Arabic newspaper articles. Our goal is to identify important entity “targets” within the post along with the polarity expressed about each target. We achieve significant improvements over multiple baselines, demonstrating that the use of specific morphological representations improves the performance of identifying both important targets and their sentiment, and that the use of distributional semantic clusters further boosts performances for these representations, especially when richer linguistic resources are not available.

1 Introduction

Target-specific sentiment analysis has recently become a popular problem in natural language processing. In interpreting social media posts, analysis needs to include more than just whether people feel positively or negatively; it also needs to include *what* they like or dislike. The task of finding all targets within the data has been called “open-domain targeted sentiment” (Mitchell et al., 2013; Zhang et al., 2015). If we could successfully identify the targets of sentiment, it would be valuable for a number of applications including sentiment summarization, question answering, understanding public opinion during political conflict, or assessing needs of populations during natural disasters.

In this paper, we address the open-domain targeted sentiment task. Input to our system consists of online posts, which can be comprised of one or multiple sentences, contain multiple entities with different sentiment, and have different

[Jealousy exists]- between [the Arab regimes]- since a long time and this is not the first time they disappoint us but this does not mess with the [Egyptians' love]+ for all the [Arab people]+ who have nothing to do with politics whatever their affiliations, and I am sure that [Egypt]+ will rise with the help of God and not [with the help of money from the Gulf]-.

[الغيرة موجودة]- بين [الأنظمة العربية]- من زمان ودي مش أول مرة يخيبوا ظننا لكن هذا لا يمس [حب المصريين]+ لكل [الشعوب العربية]+ التي لا علاقة لها بالسياسة وأيا كان انتماؤه ، وإني على يقين من أن [مصر]+ سوف تنهض بفضل الله لا [بفضل أموال الخليج]- .

Figure 1: Online post with annotated target entities and sentiment (green:pos, yellow:neg).

domains. Our goal is to identify the important entities towards which opinions are expressed in the post; these can include any nominal or noun phrase, including events, or concepts, and they are not restricted to named entities as has been the case in some previous work. The only constraint is that the entities need to be explicitly mentioned in the text. Our work also differs from much work on targeted sentiment analysis in that posts are long, complex, with many annotated targets and a lack of punctuation that is characteristic of Arabic online language. Figure 1 shows an example post, where targets are either labeled positive (green) if a positive opinion is expressed about them and negative (yellow) if a negative opinion is expressed.

To identify targets and sentiment, we develop two sequence labeling models, a target-specific model and a sentiment-specific model. Our models try to learn syntactic relations between entities and opinion words, but they also make use of (1) Arabic morphology and (2) entity semantics. Our use of morphology allows us to capture all “words” that play a role in identification of the target, while our use of entity semantics allows us to group together similar entities which may all be targets of the same sentiment; for example, if a commenter expresses negative sentiment towards

the United States, they may also express negative sentiment towards America or Obama.

Our results show that morphology matters when identifying entity targets and the sentiment expressed towards them. We find for instance that the attaching Arabic definite article *Al*+ ال is an important indicator of the presence of a target entity and splitting it off boosts recall of targets, while sentiment models perform better when less tokens are split. We also conduct a detailed analysis of errors revealing that the task generally entails hard problems such as a considerable amount of implicit sentiment and the presence of multiple targets with varying importance.

In what follows, we describe related work (§ 2), data and models (§ 3 and § 4), and linguistic decisions made for Arabic (§ 5). In § 6, we describe our use of word vector clusters learned on a large Arabic corpus. Finally, § 7 presents experiments and detailed error analysis.

2 Related Work

Aspect-based and Entity-specific Analysis

Early work in target-based sentiment looked at identifying aspects in a restricted domain: product or customer reviews. Many of these systems used unsupervised and topical methods for determining aspects of products; Hu and Liu (2004) used frequent feature mining to find noun phrase aspects, Brody and Elhadad (2010) used topic modeling to find important keywords in restaurant reviews, and Somasundaran and Wiebe (2009) mined the web to find important aspects associated with debate topics and their corresponding polarities. SemEval 2014 Task 4 (Pontiki et al., 2014) ran several subtasks for identifying aspect terms and sentiment towards aspects and terms in restaurant and laptop reviews.

Entity-specific sentiment analysis has been frequently studied in social media and online posts. Jiang et al. (2011) proposed identifying sentiment of a tweet towards a specific named entity, taking into account multiple mentions of the given entity. Biyani et al. (2015) studied sentiment towards entities in online posts, where the local part of the post that contained the entity or mentions of it was identified and the sentiment was classified using a number of linguistic features. The entities were selected beforehand and consisted of known, named entities. More recent work uses LSTM and RNN networks to determine sentiment toward aspects in product reviews (Wang et al., 2016) and

towards entities in Twitter (Dong et al., 2014; Tang et al., 2015). SemEval 2016 ran two tasks on sentiment analysis (Nakov et al., 2016) and stance (Mohammad et al., 2016) towards pre-defined topics in Twitter, both on English data.

Open domain targeted analysis In early work. Kim and Hovy (2006) proposed finding opinion target and sources in news text by automatic labeling of semantic roles. Here, opinion-target relationships were restricted to relations that can be captured using semantic roles. Ruppenhofer et al. (2008) discussed the challenges of identifying targets in open-domain text which cannot be addressed by semantic role labeling, such as implicitly conveyed sentiment, global and local targets related to the same entity, and the need for distinguishing between entity and proposition targets.

Sequence labeling models became more popular for this problem: Mitchell et al. (2013) used CRF model combinations to identify named entity targets in English and Spanish, and Yang and Cardie (2013) used joint modeling to predict opinion expressions and their source and target spans in news articles, improving over several single CRF models. Their focus was on identifying directly subjective opinion expressions (e.g. "I *hate* [this dictator]" vs. "[This dictator] is *destroying* his country.") Recent work (Deng and Wiebe, 2015) identifies entity sources and targets, as well as the sentiment expressed by and towards these entities. This work was based on probabilistic soft logic models, also with a focus on direct subjective expressions.

There is also complementary work on using neural networks for tagging open-domain targets (Zhang et al., 2015; Liu et al., 2015) in shorter posts. Previous work listed did not consider word morphology, or explicitly model distributional entity semantics as indicative of the presence of sentiment targets.

Related work in Arabic Past work in Arabic machine translation (Habash and Sadat, 2006) and named entity recognition (Benajiba et al., 2008) considered the tokenization of complex Arabic words as we do in our sequence labeling task. Analysis of such segmentation schemes has not been reported for Arabic sentiment tasks, which cover mostly sentence-level sentiment analysis and where the lemma or surface bag-of-word representations have typically been sufficient.

There are now many studies on sentence-level

sentiment analysis in Arabic news and social media (Abdul-Mageed and Diab, 2011; Mourad and Darwish, 2013; Refaee and Rieser, 2014; Salameh et al., 2015). Elarnaoty et al. (2012) proposed identifying sources of opinions in Arabic using a CRF with a number of patterns, lexical and subjectivity clues; they did not discuss morphology or syntactic relations. Al-Smadi et al. (2015) developed a dataset and built a majority baseline for finding targets in Arabic book reviews of known aspects; Obaidat et al. (2015) also developed a lexicon-based approach to improve on this baseline. Abu-Jbara et al. (2013) created a simple opinion-target system for Arabic by identifying noun phrases in polarized text; this was done intrinsically as part of an effort to identify opinion subgroups in online discussions. There are no other sentiment target studies in Arabic that we know of. In our experiments, we compare to methods similar to these baseline systems, as well as to results of English work that is comparable to ours.

Entity Clusters It has been shown consistently that semantic word clusters improve the performance of named entity recognition (Täckström et al., 2012; Zirikly and Hagiwara, 2015; Turian et al., 2010) and semantic parsing (Saleh et al., 2014); we are not aware of such work for identifying entity targets of sentiment.

3 Data

We use the Arabic Opinion Target dataset developed by Farra et al. (2015), which is publicly available¹. The data consists of 1177 online comments posted in response to *Aljazeera* Arabic newspaper articles and is part of the Qatar Arabic Language Bank (QALB) corpus (Habash et al., 2013; Zaghouani et al., 2014). The comments are 1-3 sentences long with an average length of 51 words. They were selected such that they included topics from three domains: politics, culture, and sports.

Targets are always noun phrases and they are either labeled positive if a positive opinion is expressed *about* them and negative if a negative opinion is expressed (as shown in Figure 1). Targets were identified using an incremental process where first *important* entities were identified, and then entities agreed to be neutral were discarded (the annotation does not distinguish between *neutral* and *subjective neutral*).

The data also contains ambiguous or ‘undeter-

The dictator is destroying his country

T	T	O	O	O	O
N	N	∅	∅	∅	∅

Table 1: Example of CRF annotations.

mined’ targets where annotators did agree they were targets, but did not agree on the polarity. We use these targets for training our target model, but discard them when training our sentiment polarity model. There are 4886 targets distributed as follows: 38.2% positive, 50.5% negative, and 11.3% ambiguous. We divide the dataset into a training set (80%), development set (10%), and blind test set (10%), all of which represent the three different domains. We make the splits available for researchers to run comparative experiments.

4 Sequence Labeling Models

For modeling the data, we choose Conditional Random Fields (CRF) (Lafferty et al., 2001) for the ability to engineer Arabic linguistic features and because of the success of CRF models in the past for entity identification and classification related tasks.

We build two linear chain CRF models:

1. **Target Model** This model predicts a sequence of labels \vec{E} for a sequence of input tokens \vec{x} , where

$$E_i \in \{T(target), O(not_target)\}$$

and each token x_i is represented by a feature vector \vec{f}_{i_t} . A token is labeled T if it is part of a target; a target can contain one or more consecutive tokens.

2. **Sentiment Model** This model predicts a sequence of labels \vec{S} for the sequence \vec{x} ,

$$S_i \in \{P(pos), N(neg), \emptyset(neutral)\}$$

and each token x_i is represented by a feature vector:

$$(\vec{f}_{i_s}, E_i); E_i \in \{T, O\}$$

Additionally, this model has the constraint:

$$if E_i = T, S_i \in \{P, N\}$$

and otherwise

$$S_i = \emptyset$$

¹www.cs.columbia.edu/~noura/Resources.html

The last constraint indicating that sentiment is either positive or negative is ensured by the training data, where we have no examples of target tokens having neutral sentiment. The two models are trained independently. Thus, if target words are already available for the data, the sentiment model can be run without training or running the target model. Otherwise, the sentiment model can be run on the output of the target predictor. The sentiment model uses knowledge of whether a word is a target and utilizes context from neighboring words whereby the entire sequence is optimized to predict sentiment polarities for the targets. An example sequence is shown in Table 1, where *the dictator* is an entity target towards which the writer implicitly expresses negative sentiment.

5 Arabic Morphology and Linguistics

5.1 Arabic Morphology

In Arabic, clitics and affixes can attach to the beginning and end of the word stem, making words complex. For example, in the sentence *فاستقبلوها* ‘*So they welcomed her*’, the discourse conjunction (*so* + ف), the opinion target (*her* + ها), opinion holder (*they* + او), and the opinion expression itself (*welcomed* + استقبل) are all collapsed in the same word.

Clitics, such as conjunctions +و *w+*, prepositions +ب *b+*, the definite article +ال *Al+* ‘*the*’ (all of which attach at the beginning), and possessive pronouns and object pronouns +هـ +*h* +ها +*hA* ‘*his/her*’ or ‘*him/her*’ (which attach at the end) can all function as individual words. Thus, they can be represented as separate tokens in the CRF.

The morphological analyzer MADAMIRA (Pasha et al., 2014) enables the tokenization of a word using multiple schemes. We consider the following two schemes:

- **D3**: the Declitization scheme which splits off conjunction clitics, particles and prepositions, *Al+*, and all the enclitics at the end.
- **ATB**: the Penn Arabic Treebank tokenization, which separates all clitics above except the definite article *Al+*, which it keeps attached.

For a detailed description of Arabic concatenative morphology and tokenization schemes, the reader is referred to Habash (2010).

For each token, we add a part of speech feature. For word form (non-clitic) tokens, we use the part of speech (POS) feature produced by the morphological analyzer. We consider the surface word and the lemma for representing the word form. For the clitics that were split off, we use a detailed POS feature that is also extracted from the output of the analyzer and can take such forms as *DET* for *Al+* or *poss_pron_3MP* for third person masculine possessive pronouns. Table 2 shows the words and part of speech for the input sentence *فاستقبلوها* ‘*so they welcomed her*’ *fa-istaqbalu-ha*, using the lemma representation for the word form and the D3 tokenization scheme.

These lexical and POS features are added to both our target model and sentiment model.

5.2 Sentiment Features

The choice of sentiment lexicon is an important consideration when developing systems for new and/or low-resource languages. We consider three lexicons: (1) *SIFAAT*, a manually constructed Arabic lexicon of 3982 adjectives (Abdul-Mageed and Diab, 2011), (2) *ArSenL*, an Arabic lexicon developed by linking English SentiWordNet with Arabic WordNet and an Arabic lexical database (Badaro et al., 2014), and (3) the English MPQA lexicon (Wilson et al., 2005), where we look up words by matching on the English glosses produced by the morphological analyzer MADAMIRA.

For the target model, we add token-level binary features representing subjectivity, and for the sentiment model, we add both subjectivity and polarity features.

We also add a feature specifying respectively the subjectivity or polarity of the parent word of the token in the dependency tree in the target or sentiment model.

5.3 Syntactic Dependencies

We ran the CATiB (Columbia Arabic Treebank) dependency parser (Shahrour et al., 2015) on our data. CATiB uses a number of intuitive labels specifying the token’s syntactic role: e.g. *SBJ*, *OBJ*, *MOD*, and *IDF* for the Arabic *idafa* construct (e.g. *رئيس الحكومة* *president of government*), as well as its part of speech role. In addition to the sentiment dependency features specifying the sentiment of parent words, we added dependency features specifying the syntactic role of the token in relation to its parent, and the path from the token to the parent,

Word	English	Representation	POS	Token type
<i>f</i>	so	<i>f+</i>	conj	clitic
<i>Astqblw</i>	welcomed-they	<i>isotaqobal_1</i>	verb	lemma
<i>hA</i>	her	<i>+hA</i>	ivsuff_do:3FS	clitic

Table 2: Example of morphological representation. The encoded features will be *Representation* and *POS*. The POS for *her* represents an object pronoun. The word form represented is the lemma.

e.g *nom_obj_vrb* or *nom_idf_nom*, as well as the sentiment path from the token to the parent, e.g *nom(neutral)_obj_vrb(negative)*.

5.4 Chunking and Named Entities

The morphological analyzer *MADAMIRA* also produces base phrase chunks (BPC) and named entity tags (NER) for each token. We add features for these as well, based on the hypothesis that they will help define the spans for entity targets, whether they are named entities or any noun phrases. We refer to the sentiment and target models that utilize Arabic morphology, sentiment, syntactic relations and entity chunks as **best-linguistic**.

6 Word Clusters and Entity Semantics

Similar entities which occur in the context of the same topic or the same larger entity are likely to occur as targets alongside each other and to have similar sentiment expressed towards them. They may repeat frequently in a post even if they do not explicitly or lexically refer to the same person or object. For example, someone writing about American foreign policy may frequently refer to entities such as *{the United States, America, Obama, the Americans, Westerners}*. Such entities can cluster together semantically and it is likely that a person expressing positive or negative sentiment towards one of these entities may also express the same sentiment towards the other entities in this set.

Moreover, cluster features serve as a denser feature representation with a reduced feature space compared to Arabic lexical features. Such features can benefit the CRF where a limited amount of training data is available for target entities.

To utilize the semantics of word clusters, we build word embedding vectors using the skip-gram method (Mikolov et al., 2013) and cluster them using the K-Means algorithm (MacQueen, 1967), with Euclidean distance as a metric. Euclidean distance serves as a semantic similarity metric and

has been commonly used as a distance-based measure for clustering word vectors.

The vectors are built on Arabic Wikipedia ² on a corpus of 137M words resulting in a vocabulary of 254K words. We preprocess the corpus by tokenizing (using the schemes described in section 5) and lemmatizing before building the word vectors. We vary the number of clusters and use the clusters as binary features in our target and sentiment models.

7 Experiments and Results

7.1 Experiments

Setup To build our sentiment and target models, we use CRF++ (Kudo, 2005) to build linear-chain sequences. We use a context window of +/-2 for all features except the syntactic dependencies, where we use a window of +/-4 to better capture syntactic relations in the posts. For the sentiment model, we include the context of the previous predicted label, to avoid predicting consecutive tokens with opposite polarity.

We evaluate all our experiments on the development set which contains 116 posts and 442 targets, and present a final result with the best models on the unseen test. For the SentiWordNet-based lexicon ArSenL, we tune for the sentiment score threshold and use $t=0.2$. We use Google’s word2vec tool³ for building and clustering word vectors with dimension 200. We vary the number of clusters k between 10 (25K words/cluster) and 20K (12 words/cluster).

Baselines For evaluating the **predicted targets**, we follow work in English (Deng and Wiebe, 2015) and use the *all-NP* baseline, where all nouns and noun phrases in the post are predicted as important targets.

For evaluating **sentiment towards targets**, we consider four baselines: the *majority* baseline which always predicts negative, and the *lexicon*

²<https://dumps.wikimedia.org/arwiki/20160920/arwiki-20160920-pages-articles.xml.bz2>

³<https://github.com/dav/word2vec>

baseline evaluated in the case of each of our three lexicons: manually created, WordNet-based, and English-translated. The strong lexicon baseline splits the post into sentences or phrases by punctuation, finds the phrase that contains the predicted target, and returns positive if there are more positive words than negative words, and negative otherwise. These baselines are similar to the methods of previously published work for Arabic targeted sentiment (Al-Smadi et al., 2015; Obaidat et al., 2015; Abu-Jbara et al., 2013).

We run our pipelined models for all morphological representation schemes: *surface word* (no token splits), *lemma* (no clitics), lemma with ATB clitics (contain all token splits except *Al+*), and lemma with D3 clitics (contains all token splits). We explore the effect of semantic word clusters in these scenarios. Finally we show our *best-linguistic* (high-resource) model, and the resulting integration with word clusters.

7.2 Results

Tables 3-5 show the results. Target F-measure is calculated using the *subset* metric (similar to metrics used by Yang and Cardie (2013), Irsoy and Cardie (2014)); if either the predicted or gold target tokens are a subset of the other, the match is counted when computing F-measure. Overlapping matches that are not subsets do not count (e.g. *موقف مصر* *Egypt's position* and *موقف إسرائيل* *Israel's position* do not match). For this task, in the case of multiple mentions of the same entity in the post, any mention will be considered correct if the subset matches⁴ (e.g. if *فلسطين* *Palestine* is a gold target, and *دولة فلسطين* *state of Palestine* is predicted at a different position in the post, it is still correct). This evaluation is driven from the sentiment summarization perspective: we want to predict the overall opinion in the post towards an entity.

F-pos, *F-neg*, and *Acc-sent* show the performance of the sentiment model on only the *correctly predicted* targets⁵. Since the target and sentiment models are trained separately, this is meant to give an idea of how the sentiment model would perform in standalone mode, if targets were already provided.

F-all shows the overall F-measure showing the

⁴We have also computed the performance for mention-overlap; the difference in target F-measure is 2 points and consistent across the different systems.

⁵We exclude targets with ambiguous sentiment whose polarity was not agreed on by the annotators.

performance of correctly predicted targets with correct sentiment compared to the total number of polar targets. This evaluates the end-to-end scenario of both important target and sentiment prediction.

Best results are shown in bold. Significance thresholds are calculated for the best performing systems (Tables 4-5) using the approximate randomization test (Yeh, 2000) for target recall, precision, F-measure, *Acc-sent* and *F-all*. Significance over the method in the previous row is indicated by * ($p < 0.05$), ** ($p < 0.005$), *** ($p < 0.0005$). A confidence interval of almost four F-measure points is required to obtain $p < 0.05$. Our dataset is small; nonetheless we get significant results.

Comparing Sentiment Lexicons Table 3 shows the results comparing the different baselines. All targets are retrieved using *all-NP*; sentiment is determined using the lexical baselines. As expected, the *all-NP* baseline shows near perfect recall and low precision in predicting important targets. We observe that the gloss-translated MPQA lexicon outperforms the two other Arabic lexicons among the sentiment baselines.

We believe that the hit rate of MPQA is higher than that of the smaller, manually-labeled SIFAAT, and it is more precise than the automatically generated WordNet-based lexicon ArSenL. The performance of MPQA is, however, reliant on the availability of high-quality English glosses. We found MPQA to consistently outperform in the model results, so in our *best-linguistic* models, we only show results using the MPQA lexicon.

Comparing Morphology Representations

Looking at table 4, we can see that using the lemma representation easily outperforms the sparser surface word, and that adding tokenized clitics as separate tokens outperforms representations which only use the word form. Moreover, upon using the **D3** decliticization method, we observe a significant increase in recall of targets over the **ATB** representation. This shows that the presence of the Arabic definite article *ال* *Al+* is an important indicator of a target entity; thus, even if an entity is not named, *Al+* indicates that it is a **known** entity and is likely more salient.

The more tokens are split off, the more targets are recalled, although this comes at the cost of a decrease in sentiment performance, where the lemma representation has the highest sentiment score and the D3 representation has the lowest af-

All-NP	Target			Sentiment			
	Recall	Precision	F-score	F-pos	F-neg	Acc-sent	F-all
Baseline1 Majority	98.4	29.2	45	0	72.4	56.8	12.4
Baseline2 ArSenL	98.4	29.2	45	50.6	64.3	58.6	12.7
Baseline3 SIFAAT	98.4	29.2	45	61	58	59.5	13.1
Baseline4 MPQA	98.4	29.2	45	67	63.7	65.4	14.2

Table 3: Target and sentiment results using baselines; *all-NP* for targets and lexicons for sentiment.

	Target			Sentiment			
	Recall	Precision	F-score	F-pos	F-neg	Acc-sent	F-all
Surface + POS	41	60.6	48.9	62.2	73.6	68.9	32.6
Lemma + POS	48.2**	60.5	53.7*	65.4	77.6	72.8	38.1**
+ATB tokens	52.4*	59.5	55.7	61.3	75.7	70.1	38.2
+D3 tokens	59.6**	55.7*	57.6	64.1	73	69.2	36.1

Table 4: Target and sentiment results using different morphological representations. All models use POS.

ter surface word. We believe the addition of extra tokens in the sequence (which are function words and have not much bearing on semantics) generates noise with respect to the sentiment model. All models significantly improve the baselines on F-measure; for *Acc-sent*, the surface word CRF does not significantly outperform the MPQA baseline.

Effect of Word Clusters Figures 2 - 5 show the performance of different morphological representations when varying the number of word vector clusters k . (Higher k means more clusters and fewer entities per semantic cluster.) Adding cluster features tends to further boost the recall of important targets for all morphological schemes, while more or less maintaining precision. The difference in different schemes is consistent with the results of Table 4; the D3 representation maintains the highest recall of targets, while the opposite is true for identifying sentiment towards the targets. The ATB representation shows the best overall F-measure, peaking at 41.5 using $k=250$ (compare with 38.2 using no clusters); however, it recalls much fewer targets than the D3 representation.

The effect of clusters on sentiment is less clear; it seems to benefit the D3 and ATB schemes more than lemma (significant boosts in sentiment accuracy). The improvements in F-measure and *F-all* observed by using the best value of k is statistically significant for all schemes ($k=10$ for lemma, $k=250$ for lemma+ATB, $k=500$ for lemma+D3, with *F-all* values of 40.7, 41.5, and 39.1 respectively). In general, the cluster performances tend to peak at a certain value of k which balances the

reduced sparsity of the model (fewer clusters) with the semantic closeness of entities within a cluster (more clusters).

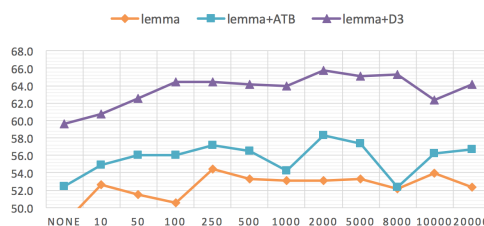


Figure 2: Target recall vs clusters.

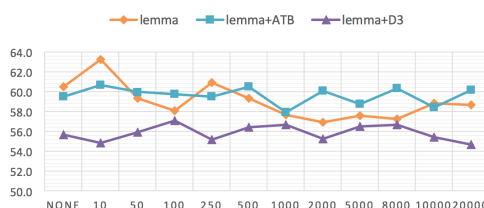


Figure 3: Target precision vs clusters.

Performance of Best Linguistic Model Table 5 shows the performance of our *best-linguistic* model, which in addition to the word form and part of speech, contains named entity and base phrase chunks, the syntactic dependency features, and the sentiment lexicon features. The best linguistic model is run using both **ATB** and **D3** tokenization schemes, and then using a combined **ATB+D3** scheme where we use D3 for the target model and remove the extra clitics before piping in the output to the sentiment model. This combined

	Target			Sentiment			
	Recall	Precision	F-score	F-pos	F-neg	Acc-sent	F-all
best-linguistic-ATB	53	62.1	57.2	68.6	79.4	75.1	40.7
best-linguistic-D3	64.2***	58.8	61.4*	62.7	75.6	70.5*	39.1
best-linguistic-D3+ATB	63.7	58.8	61.4	67.7	80	75.4***	43.1***
best-linguistic+clusters	66.2	57.8	61.8	70	80	76	44.2

Table 5: Performance of best linguistic model

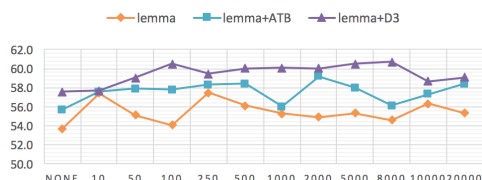


Figure 4: Target F-score vs clusters.

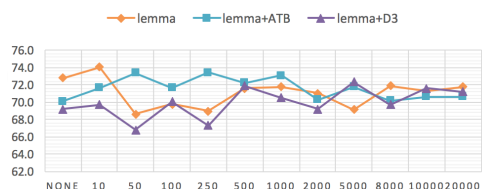


Figure 5: Sentiment accuracy vs clusters.

scheme results in the best results overall: F-score of 61.4 for targets, accuracy of 75.4 for sentiment and overall F-measure of 43.1.

Adding the richer linguistic resources results in both improved target precision, recall, and sentiment scores, with F-measure for positive targets reaching 67.7 for positive targets and 80 for negative targets. Performance exceeds that of the simpler models which use only POS and word clusters, but it is worth noting that using only the basic model with the word clusters can achieve significant boosts in recall and F-measure bringing it closer to the rich linguistic model.

The last row shows the best linguistic model **D3+ATB** combined with the clusters (best result for $k=8000$, or about 30 words per cluster). Adding the clusters improves target and F-measure scores, although this result is not statistically significant. We observe that it becomes more difficult to improve on the rich linguistic model using word clusters, which are more beneficial for low resource scenarios.

Our results are comparable to published work for most similar tasks in English: e.g. Yang and Cardie (2013) who reported target subset F-measure of ~ 65 , Pontiki et al. (2014) where best

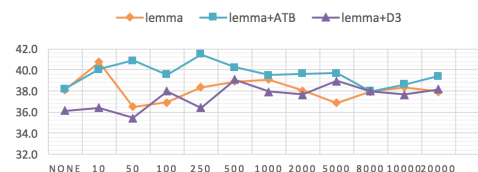


Figure 6: Overall F-score vs clusters.

	Target			Sentiment	
	R	P	F	Acc	F-all
Best-D3	63.7	52.3	57.4	69.4	35.4
Best-D3+ATB	63.7	51.8	57.1	70.3	36.8
+clusters	65.6	50.2	56.9	73.6	38.1

Table 6: Target and sentiment results on test data.

performing SemEval systems reported 70-80% for sentiment given defined aspects, and (Mitchell et al., 2013; Deng and Wiebe, 2015) for overall F-measure; we note that our tasks differ as described in section 2.

Results on blind test Table 6 shows the results on unseen test data for *best-linguistic* using **D3**, **D3+ATB** and with clusters using $k=8000$. The results are similar to what was observed in the development data.

7.3 Error Analysis

We analyzed the output of our best linguistic models on the development set, and observed the following kind of errors:

Implicit Sentiment This was the most common kind of error observed. Commenters frequently expressed complex subjective language without using sentiment words, often resorting to sarcasm, metaphor, and argumentative language. We also observed persistent errors where positive sentiment was identified towards an entity because of misleading polar words; e.g. *minds العقول* was consistently predicted to be positive even though the post in question was using implicit language to express negative sentiment; the English gloss

Example 1
Till when will [the world]- wait before it intervenes against these [crimes against humanity]- committed by this [criminal bloody regime]- which will not stop doing that... because its presence has always been associated with oppression and murder and crime... But now it's time for it to disappear and descend into [the trash of history]- .
Output the world:neg crimes:neg criminal bloody regime:neg the trash of history:neg
Example 2
[Malaysia]+ is considered the most successful country in Eastern Asia, and its economic success has spread to other [aspects of life in Malaysia]+ , for its [services to its citizens]+ have improved, and there has been an increase in [the quality of its health and educational and social and financial and touristic services]+ , which has made it excellent for foreign investments.
Output Malaysia:pos health:pos educational and social:neg financial:neg

Table 7: Good and bad examples of output by SMARTies. Gold annotations for targets are provided in the text with ‘-’ or ‘+’ reflecting negative and positive sentiment towards targets.

is *brains*, which appears as a positive subjective word in the MPQA lexicon. The posts also contained cases of complex coreference where subjective statements were at long distances from the targets they discussed.

Annotation Errors Our models often correctly predicted targets with reasonable sentiment which were not marked as important targets by annotators; this points to the subjective nature of the task.

Sentiment lexicon misses These errors resulted from mis-match between the sentiment of the English gloss and the intended Arabic meaning, leading to polar sentiment being missed.

Primary Targets The data contains multiple entity targets and not all are of equal importance. Out of the first 50 posts manually analyzed on the dev set, we found that in 38 out of 50 cases (76%) the correct *primary* targets were identified (the most important topical sentiment target(s) addressed by the post); in 4 cases, a target was predicted where the annotations contained no polar targets at all, and in the remaining cases the primary target was missed. Correct sentiment polarity was predicted for 31 out of the 38 correct targets (81.6%).

In general, our analysis showed that our system does well on posts where targets and subjective language are well formed, but that the important target identification task is difficult and made more complex by the long and repetitive nature of the posts. Table 7 shows two examples of the translated output of SMARTies, the first on more well-formed text and the second on text that is more difficult to parse.

8 Conclusions

We presented a linguistically inspired system that can recognize important entity targets along with sentiment in opinionated posts in Arabic. The targets can be any type of entity or event, and they are

not known beforehand. Both target and sentiment results significantly improve multiple lexical baselines and are comparable to previously published results in similar tasks for English, a similarly hard task. Our task is further complicated by the informal and very long sentences that are used in Arabic online posts. We showed that the choice of morphological representation significantly affects the performance of the target and sentiment models. This could shed light on further research in target-specific sentiment analysis for morphologically complex languages, an area little investigated previously. We also showed that the use of semantic clusters boosts performance for both target and sentiment identification. Furthermore, semantic clusters alone can achieve performance close to a more resource-rich linguistic model relying on syntax and sentiment lexicons, and would thus be a good approach for low-resource languages. Integrating different morphological preprocessing schemes along with clusters gives our best result.

Our code and data is publicly available⁶. Future work will consider cross-lingual clusters and morphologically different languages.

Acknowledgments

This work was supported in part by grant NPRP 6-716-1-138 from the Qatar National Research Fund, by DARPA DEFT grant FA8750-12-2-0347 and by DARPA LORELEI grant HR0011-15-2-0041. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S government. We thank anonymous reviewers for their helpful comments. We thank Yves Petinot for providing feedback on the paper. We thank Nizar Habash and Mona Diab for helpful discussions.

⁶www.cs.columbia.edu/~noura/Resources.html

References

- Muhammad Abdul-Mageed and Mona T. Diab. 2011. Subjectivity and sentiment annotation of modern standard Arabic newswire. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 110–118. Association for Computational Linguistics.
- Amjad Abu-Jbara, Ben King, Mona T. Diab, and Dragomir R. Radev. 2013. Identifying opinion subgroups in arabic online discussions. In *ACL (2)*, pages 829–835.
- Mohammad Al-Smadi, Omar Qawasmeh, Bashar Talafha, and Muhannad Quwaider. 2015. Human annotated arabic dataset of book reviews for aspect based sentiment analysis. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 726–730. IEEE.
- Gilbert Badaro, Ramy Baly, Hazem Hajj, Nizar Habash, and Wassim El-Hajj. 2014. A large scale arabic sentiment lexicon for arabic opinion mining. *ANLP 2014*, pages 165–173.
- Yassine Benajiba, Mona Diab, and Paolo Rosso. 2008. Arabic named entity recognition using optimized feature sets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 284–293. Association for Computational Linguistics.
- Prakhar Biyani, Cornelia Caragea, and Narayan Bhamidipati. 2015. Entity-specific sentiment classification of yahoo news comments. *arXiv preprint arXiv:1506.03775*.
- Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics.
- Lingjia Deng and Janyce Wiebe. 2015. Joint prediction for entity/event-level sentiment analysis using probabilistic soft logic models. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 179–189.
- Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 49–54.
- Mohamed Elarnaoty, Samir AbdelRahman, and Aly Fahmy. 2012. A machine learning approach for opinion holder extraction in arabic language. *arXiv preprint arXiv:1206.1011*.
- Noura Farra, Kathleen McKeown, and Nizar Habash. 2015. Annotating targets of opinions in arabic using crowdsourcing. In *ANLP Workshop 2015*, page 89.
- Nizar Habash and Fatiha Sadat. 2006. Arabic pre-processing schemes for statistical machine translation. *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 49–52.
- Nizar Habash, Behrang Mohit, Ossama Obeid, Kemal Oflazer, Nadi Tomeh, and Wajdi Zaghouni. 2013. QALB: Qatar Arabic language bank. In *Proceedings of Qatar Annual Research Conference (ARC-2013)*, pages ICTP-032, Doha, Qatar.
- Nizar Y. Habash. 2010. Introduction to Arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.
- Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.
- Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.
- Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.
- Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8. Association for Computational Linguistics.
- Taku Kudo. 2005. Crf++: Yet another crf toolkit. *Software available at <http://crfpp.sourceforge.net>*.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.
- Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1433–1443.
- James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

- Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654.
- Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval*, volume 16, pages 31–41.
- Ahmed Mourad and Kareem Darwish. 2013. Subjectivity and sentiment analysis of modern standard arabic and arabic microblogs. In *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 55–64.
- Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval 2016), San Diego, US (forthcoming)*, pages 1–18.
- Islam Obaidat, Rami Mohawesh, Mahmoud Al-Ayyoub, Mohammad AL-Smadi, and Yaser Jararweh. 2015. Enhancing the determination of aspect categories and their polarities in arabic reviews using lexicon-based approaches. In *Applied Electrical Engineering and Computing Technologies (AEECT), 2015 IEEE Jordan Conference on*, pages 1–6. IEEE.
- Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*, volume 14, pages 1094–1101.
- Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.
- Eshrag Refaee and Verena Rieser. 2014. Subjectivity and sentiment analysis of arabic twitter feeds with limited resources. In *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*, pages 16–21.
- Josef Ruppenhofer, Swapna Somasundaran, and Janyce Wiebe. 2008. Finding the sources and targets of subjective expressions. In *LREC*, pages 2781–2788.
- Mohammad Salameh, Saif M. Mohammad, and Svetlana Kiritchenko. 2015. Sentiment after translation: A case-study on arabic social media posts. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777.
- Iman Saleh, Alessandro Moschitti, Preslav Nakov, Lluís Màrquez, and Shafiq Joty. 2014. Semantic kernels for semantic parsing. In *EMNLP*, pages 436–442.
- Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving arabic diacritization through syntactic analysis. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1315.
- Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 226–234. Association for Computational Linguistics.
- Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 477–487. Association for Computational Linguistics.
- Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.
- Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas, November. Association for Computational Linguistics.
- Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.
- Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL (1)*, pages 1640–1649.

- Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 947–953. Association for Computational Linguistics.
- Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Os-sama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large scale arabic error annotation: Guidelines and framework. In *LREC*, pages 2362–2369.
- Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on EMNLP*, pages 612–621.
- Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. *Volume 2: Short Papers*, pages 390–396.

Exploring Convolutional Neural Networks for Sentiment Analysis of Spanish tweets

Isabel Segura-Bedmar¹, Antonio Quirós² and Paloma Martínez¹

¹Computer Science Department, Universidad Calos III de Madrid, Madrid, Spain

²s|ngular, Data & Analytics division, Madrid, Spain

{isegura, pmf}@inf.uc3m.es

{antonio.quirós}@sngular.team

Abstract

Spanish is the third-most used language on the Internet, after English and Chinese, with a total of 7.7% of Internet users (more than 277 million of users) and a huge users growth of more than 1,400%. However, most work on sentiment analysis has focused on English. This paper describes a deep learning system for Spanish sentiment analysis. To the best of our knowledge, this is the first work that explores the use of a convolutional neural network to polarity classification of Spanish tweets.

1 Introduction

Knowing the opinion of customers or users has become a priority for companies and organizations in order to improve the quality of their services and products. With the ongoing explosion of social media, it affords a significant opportunity to poll the opinion of many internet users by processing their comments. However, it should be noted that sentiment analysis, which can be defined as the automatic analysis of opinion in texts (Pang and Lee, 2008), is a challenging task because even different people often assign different polarities to a given text. Moreover, sentiment analysis can involve several Natural Language Processing (NLP) tasks such as negation or subjectivity detection, which have not been fully resolved by the NLP research community to date. On Twitter, the task is even more difficult, because the texts are small (only 140 characters) and are characterized by a informal style language utilized by users, many grammatical errors and spelling mistakes, slang and vulgar vocabulary, and plenty of abbreviations.

The shortage of training and testing data is one of the main bottlenecks for most NLP tasks in general, and for sentiment analysis in particular. This

drawback has been partially overcome thanks to the organization of shared tasks such as Sentiment Analysis in Twitter Task at SemEval 2013-2015 (Nakov et al., 2013; Rosenthal et al., 2014; Rosenthal et al., 2015; Nakov et al., 2016), which provided annotated corpora of tweets for sentiment analysis. Most research efforts in this task have focused on English texts, much less attention has been given to other languages (Abdul-Mageed et al., 2011; Kapukaranov and Nakov, 2015). However, Spanish is the third language most used on the internet, with a total of 7.7% (more than 277 million of users) and a huge internet growth of more than 1,400%.

Since its introduction in 2013, the workshop on Sentiment Analysis at SEPLN (Villena-Román et al., 2013; Villena-Román et al., 2015b; Villena-Román et al., 2015a; García Cumberras et al., 2016) has had as main goal to promote the development of methods and resources for sentiment analysis of tweets written in Spanish. In this task, the participating systems have to determine the global polarity of each tweet in the test dataset. A detailed description of the task can be found in the overview paper of TASS 2016 (García Cumberras et al., 2016).

As said above, sentiment analysis of tweets is a very challenging task because of their small size, and thereby, they usually contain very scarce contextual information. This shortage of contextual information can be supplied by exploiting knowledge from large collections of unlabelled texts. Our approach uses a convolutional neural network (CNN), which uses word embeddings as its only input. Word embeddings can be very useful for the sentiment analysis task because they are able to represent syntactic and semantic information of words (Collobert et al., 2011; Socher et al., 2013b). We do not only experiment with randomly initialized word vectors, but also explore the use of

pre-trained word embeddings. We also perform a detailed exploration of the hyper-parameters of the CNN and their effect on the results.

The paper is organized as follows. In Section 2, we discuss some related work. Section 3 describes our approach. The experimental results are presented and discussed in Section 4. We conclude in Section 5 with a summary of our findings and some directions for future work.

2 Related Work

For the past two decades, a remarkable amount of NLP research has been dedicated to the sentiment analysis task. Most of early works were focused on customer reviews of products and services, while more recently researches usually carry out the task on data from social media such as tweets and user comments. Polarity classification can be performed at three different levels: document, sentence and entity level, being the first one the one most addressed until now. However, it is increasingly demanded to know the opinion of users on specific topics (entities).

Both unsupervised and supervised approaches have been used to tackle this problem, using standard features such as unigram/bigrams, word counts or binary presence features, word position, POS tags and sentiment features from polarity lexicons such as SentiWordNet (Baccianella et al., 2010), AFFIN (Hansen et al., 2011) or iSOL (Molina-González et al., 2013). Additionally, attention has recently been directed to more complex linguistic processes such as negation and speculation detection (Pang and Lee, 2008; Cruz et al., 2015).

Only a few works have explored the use of neural networks for sentiment analysis. Socher and colleagues (2011) proposed a recursive model that is able to capture the recursive nature of sentences and learn auto-encoders for multi-word phrases. Later, they proposed a matrix-vector recursive neural network model to learn compositional vector representations for phrases and sentences of any length (Socher et al., 2012). A feed-forward neural network was designed by dos Santos and Gatti (2014) to learn relevant features from characters, words and sentences for the sentiment analysis task.

In the four editions of the TASS workshop, most systems have been based on the use of popular supervised machine learning classifiers (most

of them used SVM) and very extensive feature sets, which included lexical and morphosyntactic features (such as tokens, lemmas, n-grams, PoS tags) and sentiment features from the polarity lexicons (such as ElhPolar (Saralegi and San Vicente, 2013), iSOL (Molina-González et al., 2013) or AFFIN (Hansen et al., 2011)) to represent the information of each tweet. Only one of the systems (Vilares et al., 2015) proposed an approach based on deep learning, in particular, a neural network Long Short-Term Memory (LSTM) with a logistic function at the output layer. The evaluation of the task showed that this deep learning approach did not overcome the classical classifiers such as SVM. In TASS, there are two different evaluations: one based on 6 different polarity labels (P+, P, NEU, N, N+, NONE) and another based on just 4 labels (P, N, NEU, NONE). The state-of-art result for the task with 4 polarity levels is around 0.70 of accuracy. As expected, the best accuracy is lower (around 0.67) for the task with 6 polarity levels. A more in-depth analysis of the results and the different participating systems can be found in (Villena-Román et al., 2013; Villena-Román et al., 2015b; Villena-Román et al., 2015a; García Cumberas et al., 2016).

To the best of our knowledge, convolutional networks have not been applied to the sentiment analysis of Spanish tweets yet. Several works (dos Santos and Gatti, 2014; Severyn and Moschitti, 2015) have shown that they can be a valuable approach for English tweets, and thereby, the same could be expected also for Spanish. One of the main advantages of this architecture is that it does not require syntactic information from sentences. It should be noted that many tweets are grammatically incorrect, and thereby, those methods not based on syntactic information, could give better results.

3 Approach

3.1 The General Corpus

The General corpus was created for the TASS competition. It consists of 68,000 Spanish tweets, which were collected from November 2011 to March 2012, and covers a variety of topics such as economy, communication, politics, mass media and culture. The corpus was divided into training and test sets with a 10%-90% ratio. As said above, each tweet in the training set is classified with its polarity, which can take some of

the following values: strong positive (P+), positive (P), neutral (NEU), negative (N), strong negative (N+) and one additional for tweets without polarity (NONE). The annotation process of the training set was semi-automatic using a baseline machine learning model whose annotations were manually reviewed later by human experts. Although the test set has not been released with their gold annotations, the evaluation platform of the TASS competition is still open¹ for registered users. This platform allows participants to submit new runs and then obtain their scores.

3.2 A baseline approach

To start with, we developed a baseline based on the most common approach for polarity classification at document level: the use of very popular supervised machine learning algorithms such as SVM and logistic regression. Our goal is to compare this baseline with the CNN model.

Instead of using bag-of-words (BoW) to represent tweets, we exploited word embedding representation. A word embedding is a function to map words to low dimensional vectors, which are learned from a large collection of texts. At present, Neural Network is one of the most used learning techniques for generating word embeddings (Mikolov et al., 2013). The essential assumption of this model is that semantically close words will have similar vectors (in terms of cosine similarity). Word embeddings can help to capture semantic and syntactic relationships of the corresponding words. While the well-known BoW model involves a very large number of features (as many as the number of non-stopwords words with at least a minimum number of occurrences in the training data), the word embedding representation allows a significant reduction in the feature set size (in our case, from millions to just 300). The dimensionality reduction is a desirable goal, because it helps in avoiding over-fitting and leads to a reduction of the training and classification times, without any performance loss. Moreover, word embeddings have shown promising results in NLP tasks, such as named entity recognition (Segura-Bedmar et al., 2015), relation extraction (Alam et al., 2016), sentiment analysis (Socher et al., 2013b) or parsing (Socher et al., 2013a).

As a preprocessing step, tweets must be cleaned. First, all links, urls and usernames (these

¹www.sepln.org/workshops/tass/2016/private/evaluate.php

last ones can be easily recognized because their first character is always the symbol @) were removed. Then, the hashtags were transformed to words by removing its first character (that is, the symbol #). Taking advantage of regular expressions, the emoticons were detected and classified in order to count the number of positive and negative emoticons in each tweet and then were removed from the text. Table 1 shows the list of positive and negative emoticons, which have been taken from Wikipedia². The tweets were converted to lower-case. Moreover, the misspelled accented letters were replaced by their correct ones (for instance á with a). We also treated elongations (that is, the repetition of a character) by removing the repetition of a character after its second occurrence (for example, hoooolaaaa would be translated to hola). We also took into account laughs (for instance jajaja) which turned out to be challenging because of the diverse ways they are expressed (i.e. expressions like "ja", "jaja", jajajaja, "jiji" or jejeje and even misspelled ones like jajja-jaaj). We addressed this using regular expressions to standardize the different forms (i.e. jajjjaaj to jajaja) and then replaced them with the Spanish translation of laugh: "risa". Finally we removed all non-letters characters and all stopwords present in tweets³.

Orientation	Emoticons
Positive	:-), :), :D, :o), :], D:3, :c), :>, =], 8), =), :}, :^), :-D, 8-D, 8D, x-D, xD, X-D, XD, =-D, =D, =-3, =3, B^D, :'), :'), :*, :-*, :^*, ;-), ;), *-), *), ;-], ;], ;D, ;^), >:P, :-P, :P, X-P, x-p, xp, XP, :-p, :p, =p, :-b, :b
Negative	>:[, :-(:, (:, :-c, :-<, <:, :-[, :[, :{, ;(, :- , >:(, :^-(, :'(, D:<, D=, v.v

Table 1: List of positive and negative emoticons

Once the tweets were preprocessed, they were tokenized using the NLTK toolkit (a Python pack-

²https://en.wikipedia.org/wiki/List_of_emoticons

³<http://snowball.tartarus.org/algorithms/spanish/stop.txt>

age for NLP). To represent the tweets, we used Cardellino’s pre-trained model (Cardellino, 2016). This model is available for research community and was built from several Spanish collection texts such as Spanish Wikipedia (2015), the OPUS corpora (Tiedemann and Nygaard, 2004) or the Ancora corpus (Taulé et al., 2008), among others. It contains nearly 1.5 billion words (Cardellino, 2016). The dimension of its vectors is 300. Then, for each token, we searched its vector in the word embedding model. It should be noted that this model was trained on a collection of texts from different resources such as Spanish Wikipedia, WikiSource and Wikibooks, and none of them contains tweets. Therefore, it is possible that the main characteristics of the social media texts (such as informal style language, grammatical errors and spelling mistakes, slang and vulgar vocabulary, abbreviations, etc) are not correctly represented in this model. Indeed, we found that there was a significant number of words from the tweets (almost a 6%) that were not found in this word embedding model. We performed a review of a small sample of these words, showing that most of them were mainly hashtags.

In our baseline approach, a tweet of n tokens ($T = w_1, w_2, \dots, w_n$) is represented as the centroid of the word vectors \vec{w}_i of its tokens, as shown in the following equation:

$$\vec{T} = \frac{1}{n} \sum_{i=1}^n \vec{w}_i = \frac{\sum_{j=1}^N \vec{w}_j \cdot TF(w_j, t)}{\sum_{j=1}^N TF(w_j, t)} \quad (1)$$

where N is the vocabulary size, that is, the total number of distinct words, while $TF(w_j, t)$ refers to the number of occurrences of the j -th vocabulary word in the tweet T .

In addition to using the centroid, we completed the feature set with the following additional features:

- posWords: number of positive words present in the tweet.
- negWords: number of negative words present in the tweet.
- posEmo: number of positive emoticons present in the tweet.
- negEmo: number of negative emoticons present in the tweet.

For the posWords and negWords features we used the iSOL lexicon (Molina-González et al., 2013), a list composed by 2,509 positive words and 5,626 negative words. As described before, for the emoticons we used the listed in Table 1, but also added to the positive ones the number of laughs detected; and also, we included the number of recommendations present in the form of a “Follow Friday” hashtag (#FF), due to its ease of detection and its positive bias.

We also applied a set of emoticon’s rules as a pre-classification stage, similar to Chikersal et al. (2015), in which we determined a first stage polarity for each tweet as follows:

- If posEmo is greater than zero and negEmo is equal to zero, the tweet is marked as “P”.
- If negEmo is greater than zero and posEmo is equal to zero, the tweet is marked as “N”.
- If both posEmo and negEmo are greater than zero, the tweet is marked as “NEU”.
- If both posEmo and negEmo are equal to zero, the tweet is marked as “NONE”.

Then, the classification of tweets was performed using scikit-learn, a Python module for machine learning. This package provides many algorithms such as Random Forest, Support Vector Machine (SVM) and so on. One of its main advantages is that it is supported by extensive documentation. Moreover, it is robust, fast and easy to use. Initially, we performed experiments using three different classifiers: Random Forests, Support Vector Machines and Logistic Regression because these classifiers often achieved the best results for text classification and sentiment analysis (García Cumberas et al., 2016).

After the classification, we made three tests: i) applying no rule, ii) honoring the polarity defined by the rule, which means, we keep the predefined polarity if the tweet was marked as “P” or “N”, otherwise we take the value estimated by the classifier, and iii) a mixed approach where we give each polarity a value (N+: -2; N: -1; NEU, NONE: 0; P: 1; P+: 2) and performed an arithmetic sum of both the predefined and estimated polarity if and only if they are not equal; with that for instance, if the classifier marked a tweet as “N:-1” and the rules marked it as “P:1” the tweet will be classified as “NEU:0”.

In order to choose the best-performing classifiers, we used 10-fold cross-validation because there was no development dataset and this strategy has become the standard method in practical terms. Our experiments showed that, although the results were similar, the best settings were:

- SVM+MIX: Support Vector Machine and applying the mixed rules approach.
- LR+MIX: Logistic Regression and applying the mixed rules approach.

3.3 CNN Model

In this study, the CNN model proposed for sentiment analysis is based on the model for sentence classification described in Kim (2014). The model has been implemented using Google’s Tensorflow toolkit. Based on the fact that single level architectures seem to provide similar performance than larger networks (Kim, 2014), we decided to design our network with only a single convolutional layer, followed by a max-pooling layer and a softmax classifier as final layer. In this way, we were able to reduce the large amount of time needed to train a CNN on a large corpus as our training set (with almost 7,000 tweets).

Each tweet was represented by a matrix that concatenates the word embeddings of its words. This matrix is the input of the network. In our experiments, we learned our word embeddings from scratch, but also we tried with two different pre-trained word2vec models: Cardellino’s model, which was described above, and a pre-trained model from tweets. To train this second model of word embeddings, we used the corpus provided by the TASS organizers (68,000 tweets) as well as a very extensive collection of 8,774,487 tweets, which we collected during 2014. To do this, we used the word2vec tool, which implements the continuous bag-of-words and skip-gram architectures for computing vector representations of words (Mikolov et al., 2013). We used the continuous bag of words model with a context window of size 8. The size of the pre-trained model from tweets is 347,970 words. We randomly initialized those words that were not in the pre-trained model.

In the next layer, convolutions were performed over the word embeddings using multiple filter sizes. A filter is a sliding window of a given number of words. That is, different size windows are treated at a time. Then, a max pooling

layer extracted the most important feature (in our case, the maximum value) to reduce the computational complexity. In order to avoid over-fitting, dropout regularization was also used (Srivastava et al., 2014). This process randomly drops some units and their connections from the network during training. The final layer is a softmax prediction.

We used 10-fold cross-validation for parameter tuning. Many different combinations of hyperparameters of the neural network can be defined. Summarizing, we experimented with several variants of the model:

- CNN-rand: all words are randomly initialized and then learned during training.
- CNN-wiki: a model initialized with the word vectors from Cardellino’s pre-trained model. The word embeddings as well as the other parameters are fine-tuned for training.
- CNN-tweets: the network is initialized with the pre-trained model from tweets. As the previous model, both word embeddings and the other parameters are learned during the training.

4 Results

We adopt the same metrics used in the TASS shared task, which are the accuracy and the macro-averaged version of the precision, recall and F1.

One of our main goals is to study the effect of the word embeddings on the performance of our CNN model. Table 2 compares the models based on the type of word-embeddings used as input of the network: CNN-rand, CNN-wiki and CNN-Twitter. The other parameters of the model were set as follows: dimension of word embeddings = 300, number of filters = 128, size of filters = 3,4,5 dropout=0.5, $\lambda = 0$, batch size=64 and number of epochs=200. When the classification only considers 4 levels (POS, NEU, NEG, NONE), the best results are provided by the pre-trained model built from tweets, despite being much smaller (with less than 350,000 words) than Cardellino’s pre-trained model (with 1.5 million of words). Even the word vectors from scratch (CNN-rand) provided slightly higher performance than the CNN model trained with Cardellino’s pre-trained model. This may be due to the language style of tweets is completely different to the usual

4 polarity levels				
Approach	Acc	P	R	F1
CNN-rand	0.544	0.467	0.465	0.466
CNN-wiki	0.528	0.431	0.438	0.434
CNN-twitter	0.578	0.478	0.484	0.481

6 polarity levels				
Approach	Acc	P	R	F1
CNN-rand	0.431	0.354	0.408	0.379
CNN-wiki	0.442	0.345	0.378	0.361
CNN-twitter	0.427	0.378	0.407	0.392

Table 2: Results for 4 polarity levels (P, N, NEU and NONE) and 6 polarity levels (P+,P,N+,N,NEU and NONE)

4 polarity levels				
Approach	Acc	P	R	F1
SVM+MIX	0.652	0.506	0.510	0.508
LR+MIX	0.652	0.508	0.508	0.508
CNN-Twitter	0.637	0.518	0.519	0.518

6 polarity levels				
Approach	Acc	P	R	F1
SVM+MIX	0.527	0.411	0.449	0.429
LR+MIX	0.527	0.412	0.448	0.429
CNN-twitter	0.427	0.463	0.444	0.538

Table 3: Results of the baseline systems and the best CNN model

one of text collections (for example Wikipedia) that make up Cardellino’s corpus. As expected, the results are lower when the classification is performed using 6 polarity levels because there are less examples in the training set for the classes: P, P+ and N, N+. The pre-trained model from tweets still provides the best F1, but the best accuracy is provided by Cardellino’s pre-trained model. The worst results are obtained when the word vectors are randomly initialized.

Once we evaluated the impact of word embeddings on the performance of the CNN model, we decided to focus on the rest of its parameters. The dropout regularization is parameterized by the dropout probability $\in [0, 1]$. Our experiments using 10-fold cross validation revealed that the best performance is achieved when it is set to 0.5. As expected, several experiments showed that the larger the number of training epochs used, the more accurate the model. The final number of training epochs was set to 200. The experiments also show that the learning rate is the parameter with a largest impact in the prediction performance. The best results were obtained when this parameter was set to 3. The size of filter also seems to have a significant effect on results. Thus, the model achieved better results when our setting also considered smaller size such as 2. This may be due to tweets are small texts whose average number of words is 12 (Li et al., 2011). The best results were obtained with the filter-size parameter equals to "2,3,4,5". In order to determine the best model, we performed a comprehensive series of experiments, showing that the best parameter setting is as follows: dimension of word embeddings = 300, number of filters = 300, size of filters = 2,3,4,5 dropout=0.5, $\lambda = 3$, batch size=500 and

year	system	levels	Acc
2014	(Hurtado and Pla, 2014)	4	0.71
		6	0.64
2015	(Hurtado et al., 2015)	4	0.72
		6	0.66
2016	(Hurtado and Pla, 2016)	4	0.72
		6	0.67
	CNN-Twitter	4	0.64
		6	0.54

Table 4: Top ranking TASS systems

category	P	R	F1	Total
N	0.539	0.845	0.658	15,844
NEU	0.096	0.078	0.086	1,305
NONE	0.690	0.478	0.565	21,416
P	0.746	0.673	0.708	22,233

Table 5: CNN model’s results for each polarity (4 polarities)

number of epochs=200.

Table 3 shows the results of our baselines (using SVM or logistic regression trained with the feature set and the rules described above). It also presents the results of our best CNN model. We can observe that our CNN model provides slightly better results in terms of F1 than SVM and logistic regression. However, in terms of accuracy, these popular algorithms overcome CNN model. Meanwhile, SVM and Logistic regression provide results that are extremely similar. It is worth mentioning that the logistic regression’s performance was observably faster than the other two models. Although our CNN model worked acceptably, its performance is still far from the top ranking systems of TASS (see Table4).

Tables 5 and 6 show the scores for each polarity using the best CNN model. As expected, the lower results are obtained for those classes with less instances in the training dataset.

category	P	R	F1	Total
N	0.430	0.625	0.510	11,287
N+	0.471	0.441	0.455	4,557
NEU	0.094	0.132	0.110	1,305
NONE	0.640	0.564	0.600	21,416
P	0.119	0.501	0.193	1,488
P+	0.808	0.514	0.628	20,745

Table 6: CNN model’s results for each polarity (6 polarities)

5 Conclusion and future work

This paper explores the use of a convolutional neural network in order to extract relevant features without the necessity of handcrafted features (such as stems, named entities, PoS tags, syntactic information, etc). Our paper shows that a convolutional neural network architecture can be an effective method for sentiment analysis of Spanish tweets. Although our performance is lower than top ranking systems of the TASS workshop, our CNN model provides promising results.

As future work, we also plan to study if the inclusion of external features (such as sentiment features from polarity lexicons) to the final layer (softmax layer) achieves to improve the results. Because the General Corpus of the TASS task covers very different domains, we will perform leave-one-domain-out cross validation in order to assess the generality of our CNN model. We also plan to explore how balanced corpora and bigger datasets could help to increase the performance classification of minority classes in the training dataset. Moreover, we would like to explore other deep learning that could effectively deal with the polarity classification of Spanish tweets.

Acknowledgments

This work was supported by eGovernAbility-Access project (TIN2014-52665-C2-2-R).

References

Muhammad Abdul-Mageed, Mona T. Diab, and Mohammed Korayem. 2011. Subjectivity and sentiment analysis of modern standard arabic. In *Proceedings of the Forty-Ninth Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, volume 2, pages 587–591, Portland, Oregon, USA., June. Association for Computational Linguistics.

Firoj Alam, Anna Corazza, Alberto Lavelli, and Roberto Zanolini. 2016. A knowledge-poor approach

to chemical-disease relation extraction. *Database*, 2016:baw071.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010)*, volume 10, pages 2200–2204, Malta, May.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings.

Prerna Chikersal, Soujanya Poria, Erik Cambria, Alexander Gelbukh, and Chng Eng Siong. 2015. Modelling public sentiment in twitter: using linguistic patterns to enhance supervised learning. In *Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2015)*, pages 49–65, Cairo, Egypt, April.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Noa P. Cruz, Maite Taboada, and Ruslan Mitkov. 2015. A machine-learning approach to negation and speculation detection for sentiment analysis. *Journal of the Association for Information Science and Technology*, 67(9):2118–2136.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the Twenty-Fifth International Conference on Computational Linguistics (COLING 2014)*, pages 69–78, Dublin, Ireland., August.

Miguel Ángel García Cumbereras, Eugenio Martínez Cámara, Julio Villena Román, and Janine García Morera. 2016. Tass 2015-the evolution of the spanish opinion mining systems. *The Spanish Society for Natural Language Processing journal*, 56:33–40.

Lars Kai Hansen, Adam Arvidsson, Finn Årup Nielsen, Elanor Colleoni, and Michael Etter. 2011. Good friends, bad news-affect and virality in twitter. In *Future information technology*, pages 34–43. Springer.

Lluís-F Hurtado and Ferran Pla. 2014. Elirf-upv en tass 2014: Análisis de sentimientos, detección de tópicos y análisis de sentimientos de aspectos en twitter. In *Proceedings of TASS 2014: Workshop on Sentiment Analysis at SEPLN*, pages 75–79, Gerona, Spain., September.

Lluís-F Hurtado and Ferran Pla. 2016. Elirf-upv en tass 2016: Análisis de sentimientos en twitter. In *Proceedings of TASS 2016: Workshop on Sentiment Analysis at SEPLN*, pages 35–40, Salamanca, Spain., September.

- Lluís-F Hurtado, Ferran Pla, and Davide Buscaldi. 2015. Elirf-upv en tass 2015: Análisis de sentimientos en twitter. In *Proceedings of TASS 2015: Workshop on Sentiment Analysis at SEPLN*, pages 35–40, Alicante, Spain., September.
- Borislav Kapukaranov and Preslav Nakov. 2015. Fine-grained sentiment analysis for movie reviews in bulgarian. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 266–274, Hissar, Bulgaria, September. INCOMA Ltd. Shoumen, BULGARIA.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.
- Baichuan Li, Xiance Si, Michael R Lyu, Irwin King, and Edward Y Chang. 2011. Question identification on twitter. In *Proceedings of the Twentieth ACM international conference on Information and knowledge management (CIKM 2011)*, pages 2477–2480, Glasgow, UK., October.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Twenty-Seventh Conference on Neural Information Processing Systems (NIPS 2013)*, volume 26, pages 5–10, December.
- M. Dolores Molina-González, Eugenio Martínez-Cámara, María-Teresa Martín-Valdivia, and José M. Perea-Ortega. 2013. Semantic orientation for polarity classification in spanish reviews. *Expert Systems with Applications*, 40(18):7250–7257.
- Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June. Association for Computational Linguistics.
- Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California, June. Association for Computational Linguistics.
- Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.
- Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 73–80, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 451–463, Denver, Colorado, June. Association for Computational Linguistics.
- Xavier Saralegi and Iaki San Vicente. 2013. Elhuyar at TASS 2013. In *Proceedings of the Workshop on Sentiment Analysis at SEPLN (TASS 2013)*, pages 143–150, Madrid, Spain., September.
- Isabel Segura-Bedmar, Víctor Suárez-Paniagua, and Paloma Martínez. 2015. Exploring word embedding for drug name recognition. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 64–72, Lisbon, Portugal, September. Association for Computational Linguistics.
- Aliaksei Severyn and Alessandro Moschitti. 2015. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 464–469, Denver, Colorado, June. Association for Computational Linguistics.
- Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.
- Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013a. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on*

Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. Ancora: Multilevel annotated corpora for catalan and spanish. In *Proceedings of the Sixth Language Resources and Evaluation Conference (LREC 2008)*, pages 96–101, Marrakech, Morocco., May.

Jörg Tiedemann and Lars Nygaard. 2004. The opus corpus-parallel and free: <http://logos.uio.no/opus>. In *Proceedings of the Second Language Resources and Evaluation Conference (LREC 2004)*, pages 1183–1186, Lisbon, Portugal., May.

David Vilares, Yeraí Doval, Miguel A Alonso, and Carlos Gómez-Rodríguez. 2015. Lys at tass 2015: Deep learning experiments for sentiment analysis on spanish tweets. In *Proceedings of TASS 2015: Workshop on Sentiment Analysis at SEPLN*, pages 47–52, Alicante, Spain., September.

Julio Villena-Román, Sara Lana Serrano, Eugenio Martínez Cámara, and José Carlos González-Cristóbal. 2013. Tass-workshop on sentiment analysis at sepln. *The Spanish Society for Natural Language Processing journal*, 50:37–44.

Julio Villena-Román, Janine García-Morera, Miguel A. García-Cumbreras, Eugenio Martínez-Cámara, M. Teresa Martín-Valdivia, and L. Alfonso Urena-López. 2015a. Overview of tass 2015. In *Proceedings of the TASS 2015 Workshop on Sentiment Analysis at SEPLN*, volume 1397, pages 13–21, Alicant, Spain, September.

Julio Villena-Román, Eugenio Martínez-Cámara, Janine García-Morera, and Salud M. Jiménez-Zafra. 2015b. Tass 2014-the challenge of aspect-based sentiment analysis. *The Spanish Society for Natural Language Processing journal*, 54:61–68.

Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: A Generative Approach to Sentiment Analysis

Amr El-Desoky Mousa¹ and Björn Schuller^{1,2}

¹Chair of Complex & Intelligent Systems, University of Passau, Passau, Germany

²Department of Computing, Imperial College London, London, UK

amr.mousa@uni-passau.de

schuller@ieee.org

Abstract

Traditional learning-based approaches to sentiment analysis of written text use the concept of bag-of-words or bag-of- n -grams, where a document is viewed as a set of terms or short combinations of terms disregarding grammar rules or word order. Novel approaches de-emphasize this concept and view the problem as a sequence classification problem. In this context, recurrent neural networks (RNNs) have achieved significant success. The idea is to use RNNs as discriminative binary classifiers to predict a positive or negative sentiment label at every word position then perform a type of pooling to get a sentence-level polarity. Here, we investigate a novel generative approach in which a separate probability distribution is estimated for every sentiment using language models (LMs) based on long short-term memory (LSTM) RNNs. We introduce a novel type of LM using a modified version of bidirectional LSTM (BLSTM) called contextual BLSTM (cBLSTM), where the probability of a word is estimated based on its full left and right contexts. Our approach is compared with a BLSTM binary classifier. Significant improvements are observed in classifying the IMDB movie review dataset. Further improvements are achieved via model combination.

1 Introduction

Sentiment analysis of text (also known as opinion mining) is the process of computationally identifying and categorizing opinions expressed in a piece of text in order to determine the writer's attitude towards a particular topic. Due to the tremendous

increase in web content, many organizations became increasingly interested in analyzing this big data in order to monitor the public opinion and assist decision making. Therefore, sentiment analysis attracted the interest of many researchers.

The task of sentiment analysis can be seen as a text classification problem. Depending on the target of the analysis, the classes can be described by continuous primitives such as valence, a polarity state (positive or negative attitude), or a subjectivity state (objective or subjective). In this work, we are interested in the binary classification of documents into a positive or negative attitude. Such detection of polarity is a non-trivial problem due to the existence of noise, comparisons, vocabulary changes, and the use of idioms, irony, and domain specific terminology (Schuller et al., 2015).

Traditional approaches to sentiment analysis rely on the concept of bag-of-words or bag-of- n -grams, where a document is viewed as a set of terms or short combinations of terms disregarding grammar rules or word order. In this case, usually, the analysis involves: tokenization and parsing of text documents, careful selection of important features (terms), dimensionality reduction, and classification of the documents into categories. For example, Pang et al. (2002) have considered different classifiers, such as Naive Bayes (NB), maximum entropy (MaxEnt), and support vector machines (SVM) to detect the polarity of movie reviews. Pang and Lee (2004) have combined polarity and subjectivity analysis and proposed a technique to filter out objective sentences of movie reviews based on finding minimum cuts in graphs. In (Taboada et al., 2011; Ding et al., 2008), lexicon-based techniques are examined, where word-level sentimental orientation scores are used to evaluate the polarity of product reviews. More advanced approaches utilize word or n -gram vectors, like in (Maas et al., 2011; Dahl et al., 2012).

Novel approaches are mainly based on artificial neural networks (ANNs). These approaches de-emphasize the concept of bag-of-words or bag-of- n -grams. A document is viewed as a set of sentences, each sentence is a sequence of words. The sentiment problem is rather considered as a sequence classification problem. For example, in (Dong et al., 2014; Dong et al., 2016), RNN classifiers are used with an adaptive method to select relevant semantic composition functions for obtaining vector representations of sentences. This is found to improve sentiment classification on the Stanford Sentiment Treebank (SST). Rong et al. (2014) have used a RNN model to learn word representation simultaneously with the sentiment distribution. Santos and Gatti (2014) have proposed a convolutional neural network (CNN) that exploits from character- to sentence-level information to perform sentiment analysis on the Stanford Twitter Sentiment (STS) corpus. Kalchbrenner et al. (2014) have used a dynamic convolutional neural network (DCNN) with a dynamic k -max pooling to perform sentiment analysis on the SST and Twitter sentiment datasets. Lai et al. (2015) have utilized a combination of RNNs and CNNs called recurrent convolutional neural network (RCNN) to perform text classification on multiple datasets including sentiment analysis on the SST dataset.

Other novel approaches use tree structured neural models instead of sequential models (like RNNs) in order to capture complex semantic relationships that relate words to phrases. Despite their good performance, these models rely on existing parse trees of the underlying sentences which are, in most cases, not readily available or not trivial to generate. For example, Socher et al. (2013) have introduced a recursive neural tensor network (RNTN) to predict the compositional semantic effects in the SST dataset. In (Tai et al., 2015; Le and Zuidema, 2015), tree-structured LSTMs are used to improve the earlier models.

Another perspective to the sentiment problem is to assume that each sentence with a positive or negative class is drawn from a particular probability distribution related to that class. Then, instead of estimating a discriminative model that learns how to separate sentiment classes in sentence space, we estimate a generative model that tells us how these sentences are generated. This generative approach can be better or complementary in some sense to the discriminative approach.

The probability distributions over word sequences are well known as language models (LMs). They have also been used for sentiment analysis. However, no trial is made to go beyond simple bigram models. For example, Hu et al. (2007b) have estimated two separate positive and negative LMs from training collections. Tests are performed by computing the Kullback-Leibler divergence between the LM estimated from the test document and the sentiment LMs. Therein, uni- and bigram models are shown to outperform SVM models in classifying a movie review dataset. In (Hu et al., 2007a), a batch of terms in a domain are identified. Then, two different unigram LMs representing classifying knowledge for every term are built up from subjective sentences. A classifying function based on the generation of a test document is defined for the sentiment classification. This approach has outperformed SVM on a Chinese digital product review dataset. Liu et al. (2012) have employed an emoticon smoothed unigram LM to perform sentiment classification.

In this paper, we compare the generative LM approach with the discriminative binary classification approach. We estimate a separate probability distribution for each sentiment using long-span LMs based on unidirectional LSTMs (Sundermeyer et al., 2012) trained to predict a word depending on its full left context. The probability scores from positive and negative LMs are used to classify unseen sentences. In addition, we introduce a novel type of LM using a modified version of the standard bidirectional LSTM called contextual bidirectional LSTM (cBLSTM). In contrast to the unidirectional model, this model is trained to predict a word depending on its full left and right contexts. Moreover, we combine the LM approach with the binary classification approach using linear interpolation of probabilities. We observe that the cBLSTM LM outperforms both the LSTM LM and the BLSTM binary classifier. Combining approaches together yields further improvements. Models are evaluated on the IMDB large movie review dataset¹ (Maas et al., 2011).

2 Language Models

A statistical LM is a probability distribution over word sequences that assigns a probability $p(w_1^M)$ to any word sequence w_1^M of length M . Thus, it provides a way to estimate the relative likelihood

¹<http://ai.stanford.edu/~amaas/data/sentiment/>

of different phrases. It is a widely used model in many natural language processing tasks, like automatic speech recognition, machine translation, and information retrieval. Usually, to estimate a LM, the assumption of the $(n - 1)^{th}$ order Markov process is used (Bahl et al., 1983), in which a current word w_m is assumed conditionally dependent on the preceding $(n - 1)$ history words, such that:

$$p(w_1^M) \approx \prod_{m=1}^M p(w_m | w_{m-n+1}^{m-1}). \quad (1)$$

This is called an n -gram LM. A conventional approach to estimate these probabilities is the back-off LM which is based on count statistics collected from the training text. In addition to the initial n -gram approximation, a major drawback of this model is that it backs-off to a shorter history whenever insufficient statistics are observed for a given n -gram. Novel state-of-the-art LMs are based on ANNs like RNNs that provide long-span probabilities conditioned on all predecessor words (Mikolov et al., 2010; Kombrink et al., 2011).

3 Unidirectional RNN Models

3.1 Standard RNN

A RNN maps from a sequence of input observations to a sequence of output labels. The mapping is defined by a set of activation weights and a non-linear activation function. Recurrent connections allow to access activations from past time steps. For an input sequence x_1^T , a RNN computes the hidden sequence h_1^T and the output sequence y_1^T by performing the following operations for time steps $t = 1$ to T (Graves et al., 2013):

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2)$$

$$y_t = W_{hy}h_t + b_y, \quad (3)$$

where \mathcal{H} is the hidden layer activation function, W_{xh} is the weight matrix between the input and hidden layer, W_{hh} is the recurrent weight matrix between the hidden layer and itself, W_{hy} is the weight matrix between the hidden and output layer, b_h and b_y are the hidden and output layer bias vectors respectively. \mathcal{H} is usually an element-wise application of the sigmoid function.

3.2 LSTM RNN

In (Hochreiter and Schmidhuber, 1997), an alternative RNN called Long Short-Term Memory (LSTM) is introduced where the conventional neuron is replaced with a so-called *memory cell* controlled by input, output and forget gates in order to

overcome the vanishing gradient problem of traditional RNNs. In this case, \mathcal{H} can be described by the following composite function:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (4)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (5)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (7)$$

$$h_t = o_t \tanh(c_t), \quad (8)$$

where σ is the sigmoid function, i, f, o , and c are respectively the input, forget, output gates, and cell activation vectors (Graves et al., 2013).

3.3 LSTM LM

In a LSTM LM, the time steps correspond to the word positions in a training sentence. At every time step, the network takes as input the word at the current position encoded as a 1-hot binary vector. The input vector is then passed to one or more recurrent hidden layers with self connections that implicitly take into account all the previous history words presented to the network. The output of the final hidden layer is passed to an output layer with a softmax activation function to produce a correctly normalized probability distribution. The target output at each word position is the next word in the sentence. A cross-entropy loss function is used which is equivalent to maximizing the likelihood of the training data. At the end, the network can predict the long-span conditional probability $p(w_m | w_1^{m-1})$ for any word $w_m \in V$ and a given history w_1^{m-1} , where V is the vocabulary. Fig. 1 shows an unfolded example of a LSTM LM over a sentence $\langle s \rangle w_1 w_2 w_3 \langle /s \rangle$, where $\langle s \rangle$ and $\langle /s \rangle$ are the sentence start and end symbols.

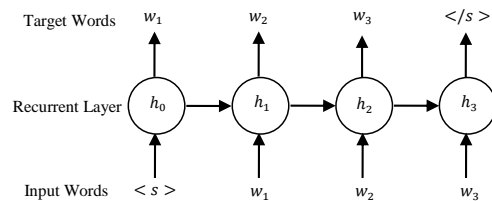


Figure 1: Architecture of a LSTM LM predicting a word given its full previous history.

4 Bidirectional RNN Models

4.1 BLSTM RNN

A BLSTM processes input sequences in both directions with two sub-layers in order to account for the full input context. These two sub-layers

compute forward and backward hidden sequences \vec{h} , \overleftarrow{h} respectively, which are then combined to compute the output sequence y (see Fig. 2), thus:

$$\vec{h}_t = \mathcal{H}(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}\vec{h}_{t-1} + b_{\vec{h}}) \quad (9)$$

$$\overleftarrow{h}_t = \mathcal{H}(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \quad (10)$$

$$y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \quad (11)$$

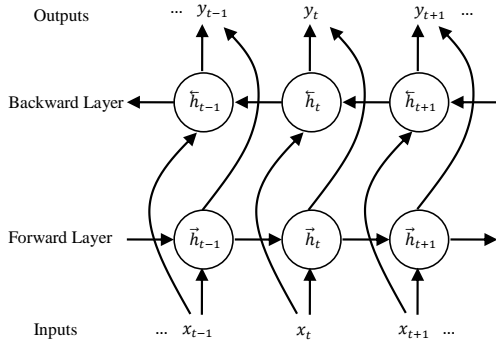


Figure 2: Architecture of a BLSTM, every output depends on the whole input sequence.

4.2 Contextual BLSTM LM

The standard BLSTM described in Section 4.1 is not suitable for estimating LMs. This is because it predicts every output symbol depending on the whole input sequence. Since a LM indeed uses the same word sequence in both input and target sides of the network, it would be incorrect to predict a word given the whole input sentence. Rather, it is required to predict a word given the full left and right context while excluding the predicted word itself from the conditional dependence. To allow for this, we modify the architecture of the standard BLSTM such that it accounts for a contextual dependence rather than a full sequence dependence. The new model is called a contextual BLSTM or cBLSTM in short. The architecture of this model is illustrated in Fig. 3.

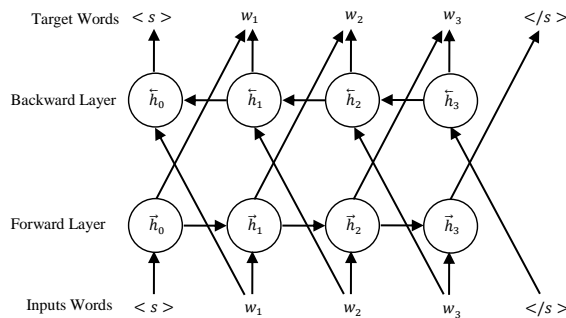


Figure 3: Architecture of a cBLSTM LM predicting a word given its full left and right contexts.

The model consists of a forward and a backward sub-layer. The forward sub-layer receives the encoded input words starting from the sentence start symbol up to the last word before the sentence end symbol (sequence $\langle s \rangle w_1 w_2 w_3$ in Fig. 3). The forward hidden states are used to predict words starting from the first word after the sentence start symbol up to the sentence end symbol (sequence $w_1 w_2 w_3 \langle /s \rangle$ in Fig. 3). The backward sub-layer does exactly the reverse operation. The two sub-layers are interleaved together in order to adjust the conditional dependence such that the prediction of any target word depends on the full left and right contexts. Note that the hidden state at the first as well as the last time step needs to be padded by zeros so that the size of the hidden vector is consistent across all time steps. At the end, the model can effectively predict the conditional probability $p(w_m | w_1^{m-1}, w_{m+1}^M)$ for any word $w_m \in V$, left context w_1^{m-1} and right context w_{m+1}^M , where V is the vocabulary and M is the length of the sentence. Table 1 shows the predicted probability at each time step of Fig. 3. Note that one direction dependence is maintained at the start and end of sentence (time steps 1 and 5).

time step	predicted conditional prob.
1	$p(\langle s \rangle w_1 w_2 w_3 \langle /s \rangle)$
2	$p(w_1 \langle s \rangle, w_2 w_3 \langle /s \rangle)$
3	$p(w_2 \langle s \rangle w_1, w_3 \langle /s \rangle)$
4	$p(w_3 \langle s \rangle w_1 w_2, \langle /s \rangle)$
5	$p(\langle /s \rangle \langle s \rangle w_1 w_2 w_3)$

Table 1: Predicted conditional probabilities at every time step of the cBLSTM shown in Fig. 3.

Our implementation of the novel cBLSTM RNN model is integrated into our publicly available CURRENNT² toolkit initially introduced by Weninger et al. (2014). A new version of the toolkit with the novel implementations is planned to be available by the date of publication.

Here, it is worth noting that deep cBLSTM models can not be easily constructed by stacking multiple hidden bidirectional layers together. The reason is that the hidden states obtained after the first bidirectional layer are dependent on the full left and right contexts. If these states are utilized as inputs to a second bidirectional layer that identically repeats the same operation again, then the desired conditional dependence will not be correctly

²<http://sourceforge.net/p/currentnt>

maintained. One method to solve this problem is to create deeper models by stacking multiple forward and backward sub-layers independently. The fusion of both sub-layers takes place at the end of the deep stack. The implementation of such a deep cBLSTM model is not yet available.

5 Dataset

Our experiments on sentiment analysis are performed on the IMDB large movie review dataset (v1.0) introduced by Maas et al. (2011). The labeled partition of the dataset consists of 50k balanced full-length movie reviews with 25k positive and 25k negative reviews extracted from the Internet Movie Database (IMDB)³.

Since the reviews are in a form of long paragraphs which are difficult to handle directly with RNNs, we break down these paragraphs into relatively short sentences based on punctuation clues. After breaking down the paragraphs, the average number of sentences per review is around 13 sentences. We randomly selected 1000 positive and 1000 negative reviews as our test set. A similar number of random reviews are selected as a development set. The remaining reviews are used as a training set. Note that this is not the official dataset division provided by Maas et al. (2011), where 25k balanced reviews are dedicated for training and the other 25k balanced reviews are dedicated for testing. The reasons not to follow the official division are firstly that, it does not provide a development set; secondly, our proposed models need much data to train well as revealed by initial experiments; thirdly, it would be very time consuming to use the whole data as one partition and perform multi-fold cross validation as usually adopted with large sentiment datasets (Schuller et al., 2015). A preprocessed version of the IMDB dataset with the modified partitioning is planned to be available for download by the date of publication.

A word list of the 10k most frequent words is selected as our vocabulary. This covers around 95% of the words in our development and test sets. Any out-of-vocabulary word is mapped to a special *unk* symbol. We use the classification accuracy as our evaluation measure. The unweighted average F1 scores over positive and negative classes are also calculated. However, their values are found almost similar to the classification accuracies. Therefore, only classification accuracies are reported.

³<http://www.imdb.com>

6 Related Work

The work of this paper is closely related to several previous publications that report sentiment classification accuracy on the same dataset. For example, in (Maas et al., 2011), the IMDB dataset is introduced and a semi-supervised word vector induction framework is used, where an unsupervised probabilistic model similar to latent Dirichlet allocation (LDA) is proposed to learn word vectors. Another supervised model is utilized to constrain words expressing similar sentiment to have similar representations in vector space. In (Dahl et al., 2012), documents are treated as bags of n -grams. Restricted Boltzmann machines (RBMs) are used to extract vector representations for n -grams. Then, a linear SVM model is utilized to classify documents based on the resulting feature vectors. Wang and Manning (2012) have used a variant of SVM with Naive Bayes log-count ratios as well as word bigrams as features. This modified SVM model is referred to as NBSVM. In our previous publication (Schuller et al., 2015), LSTM LMs trained on 40% of the whole IMDB dataset are used for performing sentiment analysis. However, a carefully tuned MaxEnt classifier is found to perform better. Le and Mikolov (2014) have used a paragraph vector methodology with an unsupervised algorithm based on feed-forward neural networks that learns fixed-length vector representations from variable-length texts. All these publications use the official IMDB dataset division except for (Schuller et al., 2015), where a similar division as in this paper is used. To give a comprehensive idea about the aforementioned techniques, we show in Table 2 the classification results as reported in the related publications. Note that only the results of (Schuller et al., 2015) are directly comparable to our results.

experiment	Accuracy [%]
Maas et al. (2011)	88.89
Dahl et al. (2012)	89.23
Wang and Manning (2012)	91.22
Schuller et al. (2015)*	91.55
Le and Mikolov (2014)	92.58

Table 2: Sentiment classification accuracies from previous publications on the IMDB dataset.

In relation to our novel cBLSTM LM, previous trials have been made to estimate bidirectional

LMs. For example, in (Frinken et al., 2012), distinct forward and backward LMs are estimated for handwriting recognition. However, no trial is made to go beyond 4-gram models. In (Xiong et al., 2016), standard forward and backward RNN LMs are separately estimated for a conversational speech recognition task. The log probabilities from both models are added. In (Arisoy et al., 2015), bidirectional RNNs and LSTMs are used to estimate LMs for an English speech recognition task. Therein, the standard bidirectional architecture (as in Fig. 2) is used without modifications. This causes circular dependencies to arise when combining probabilities from multiple time steps. Therefore, pseudo-likelihoods are utilized rather than true likelihoods which is not perfect from the mathematical point of view. Not surprisingly, the BLSTM LMs do not yield any gain over the LSTM LMs. In addition, the perplexity of such a model becomes invalid. More importantly, in (Peris and Casacuberta, 2015), bidirectional RNN LMs are used for a statistical machine translation task. However, only standard RNNs but not LSTMs are utilized. Furthermore, no details are provided about how the model is exactly modified and how the left and right dependencies are maintained over time steps.

7 Sentiment Classification

7.1 Generative LM-based classifier

Our first approach to sentiment classification is the generative approach based on LMs. We either use LSTM LMs described in Section 3.3 or cBLSTM LMs described Section 4.2. Two separate LMs are estimated from positive and negative training data. We use networks with a single hidden layer that consists of 300 memory cells followed by a softmax layer with a dimension of $10k + 3$. This is equal to the full vocabulary size in addition to $\langle s \rangle$, $\langle /s \rangle$, and unk symbols representing sentence start, sentence end, and unknown word symbols respectively. In case of using cBLSTM networks, a single hidden layer of 600 memory cells is used (300 cells for each forward and backward sub-layer). A cross-entropy loss function is used with a momentum of 0.9. We use sentence-level mini-batches of size 100 sentences computed in parallel. The learning rate is set initially to 10^{-3} and then decreased gradually to 10^{-6} . The training process is controlled by monitoring the cross-entropy error over the development set.

In addition, we use a data sub-sampling methodology during training. For this purpose, a traditional 5-gram backoff LM is created out of the development data, we call this a *ranking LM*. Then, all training sentences are ranked according to their perplexities with the ranking LM. Using these ranks, we divide our training sentences into three partitions that reflect the relative importance of the data, such that the first partition contains the 100k sentences with the lowest perplexities, the second partition contains the 100k sentences with next lowest perplexities. The third partition contains all the other sentences. Instead of using the whole training data in each epoch, we use a random sample with more sentences from the first two partitions than the third one. After a sufficient number of epochs, the whole training data is covered. The sub-sampling approach speeds up the training and makes it feasible with any size of training data. At the same time, the training is focused on the relatively more important examples. In addition, it adds a useful regularization to the training process. Yet, it leads to a less smoother convergence. To show the efficiency of our sentence ranking methodology, Table 3 shows examples of the highest and lowest ranked sentences from positive and negative training data.

most +ve	this is one of the best films ever made.
least +ve	cheap laughs but great value.
most -ve	this is one of the worst movies i have ever seen.
least -ve	life's too short.

Table 3: Examples of the highest/lowest ranked sentences from positive/negative training data.

After training the neural networks, each of the positive and negative sentiment LM estimates a probability distribution for the corresponding sentiment, we call these probability distributions p_+ and p_- . To evaluate the sentiment of some test review, we calculate the perplexity of each model p_+ and p_- with respect to the whole review. Thus, given a probability distribution p , and a review text S composed of K sentences $S = s_1, \dots, s_K$, each sentence $s_k : 1 \leq k \leq K$ is composed of a sequence of M_k words $s_k = w_1^k, w_2^k, \dots, w_{M_k}^k$; we calculate the perplexity $PP_p(S)$ of a model p with respect to text S . It is a very common measure-

ment of how well a probability distribution predicts a sample. A low perplexity indicates that the probability distribution is good at predicting the sample. Perplexity is defined as the exponentiated negative average log-likelihood, or in other words, the inverse of the geometric average probability assigned by the model to each word in the sample. We calculate the Perplexity using Equation 12 if the model p is based on LSTM, and using Equation 13 if the model is based on cBLSTM:

$$PP_p(S) = \left[\prod_{k=1}^K \prod_{m=1}^{M_k} p(w_m^k | w_1^k, w_2^k, \dots, w_{m-1}^k) \right]^{\frac{-1}{N}} \quad (12)$$

$$PP_p(S) = \left[\prod_{k=1}^K \prod_{m=1}^{M_k} p(w_m^k | w_1^k, w_2^k, \dots, w_{m-1}^k; w_{m+1}^k, w_{m+2}^k, \dots, w_{M_k}^k) \right]^{\frac{-1}{N}}, \quad (13)$$

where $N = \sum_{k=1}^K M_k$ is the total number of words in text S . Then, a sentiment polarity $\mathcal{P} \in \{-1, +1\}$ is assigned to S according to the following decision rule:

$$\mathcal{P}(S) = \begin{cases} +1 & \text{if } PP_{p_+}(S) < PP_{p_-}(S) \\ -1 & \text{otherwise} \end{cases}. \quad (14)$$

7.2 Discriminative BLSTM-based Binary Classifier

Our second approach to sentiment classification is the discriminative approach based on BLSTM RNNs described in Section 4.1. We use BLSTM networks with a single hidden layer that consists of 600 memory cells (300 cells for each forward and backward sub-layer). Since the BLSTM performs a binary classification task, only a single output neuron is used with a sigmoid activation function. A cross-entropy loss function is used with a momentum of 0.9. The same training settings like the case of LSTM/cBLSTM LMs are used including sub-sampling with the same partitioning of the training data. However, a single training dataset with all positive and negative reviews is used. For a sentence with a positive sentiment, the target outputs are set to *ones* at all time steps. For a sentence with a negative sentiment, the target outputs are set to *zeros* at all time steps. Since the sigmoid function provides output values in the interval $[0,1]$,

the network is trained to produce the probability of the positive class at every time step. Although the output of the BLSTM network at a given time step is dependent on the whole input sequence, it is widely known that every output is more affected by the inputs at closer time steps in both directions. Therefore, a sentence-level sentiment can be deduced by comparing the average probability mass assigned to the positive class over all time steps with the average probability mass assigned to the negative class. Thus, similar to Section 7.1, given a review text S composed of K sentences, each sentence is a sequence of M_k words, we calculate two probabilities $p_+(S)$ and $p_-(S)$ that the review S has a positive or negative sentiment using Equations 15 and 16 respectively:

$$p_+(S) = \frac{1}{N} \sum_{k=1}^K \sum_{m=1}^{M_k} p_+(w_m^k) \quad (15)$$

$$p_-(S) = \frac{1}{N} \sum_{k=1}^K \sum_{m=1}^{M_k} (1 - p_+(w_m^k)), \quad (16)$$

where N is the total number of words in text S , and $p_+(w_m^k)$ is the probability that a positive class is assigned to the word at position m of the k^{th} sentence of the review S . Then, a sentiment polarity $\mathcal{P} \in \{-1, +1\}$ is assigned to S according to the following decision rule:

$$\mathcal{P}(S) = \begin{cases} +1 & \text{if } p_+(S) > p_-(S) \\ -1 & \text{otherwise} \end{cases}. \quad (17)$$

7.3 Model Combination

The probability scores of the generative LM-based classifier and the discriminative BLSTM-based binary classifier discussed in Sections 7.1 and 7.2 can be combined together via linear interpolation. This is achieved by first normalizing the probabilities from the LMs such that the probabilities of positive and negative classes for a given review are summed up to 1.0. Note that this normalization property holds by default for the BLSTM-based binary classifier. Then, the probabilities of both models are linearly interpolated to obtain a single probability score. The interpolation weights are optimized on the development data.

8 Experimental Results

Table 4 shows the results of our experiments. All the neural networks in this work are trained

and optimized using our own CURRENNT toolkit (Weninger et al., 2014). Both the LSTM and cBLSTM LMs are linearly interpolated with two additional LMs, namely a 5-gram backoff LM smoothed with modified Kneser-Ney smoothing (Kneser and Ney, 1995), and another 5-gram MaxEnt LM (Alumäe and Kurimo, 2010). These two models are estimated using the SRILM language modeling toolkit (Stolcke, 2002).

classification model	Acc. [%]
LSTM LM	89.58
+ 5-grm backoff LM	91.05
+ 5-grm MaxEnt LM	91.23
cBLSTM LM	89.88
+ 5-grm backoff LM	91.38
+ 5-grm MaxEnt LM	91.48
BLSTM binary classifier	90.15
LSTM LM + BLSTM binary classifier	92.35
cBLSTM LM + BLSTM binary classifier	92.83
Schuller et al. (2015) LSTM + 5-grm LM	90.50
Schuller et al. (2015) MaxEnt classifier	91.55

Table 4: Sentiment classification accuracies measured on the IMDB dataset.

We observe that the use of cBLSTM LM as a generative sentiment classifier significantly outperforms the use of both LSTM LM and BLSTM discriminative binary classifiers. The statistical significance is verified using a bootstrap method of significance analysis described by Bisani and Ney (2004). The probability of improvement (POI_{boot}) is around 95%. Combining LM-based classifiers with BLSTM-based binary classifiers via linear interpolation of probabilities achieves further improvements. Our best accuracy (92.83%) is obtained by combining the cBLSTM LM classifier with the BLSTM binary classifier. These results reveal that both the generative and discriminative approaches are complementary in solving the sentiment classification problem.

Finally, our best result is better than the best previously published result in (Schuller et al., 2015) on the same IMDB dataset with the same dataset partitioning. Even though they are not directly comparable, our results are better than other previously published results reported in Table 2 where a different dataset partitioning is used.

For illustration, Table 5 shows two examples of positive and negative reviews that could not be

correctly classified by the discriminative BLSTM binary classifier, however they are correctly classified by the cBLSTM LM classifier. We can observe the implicit indication of the writer’s attitude towards the movie which can not be easily captured by simple approaches. In this case, learning a separate long-span bidirectional probability distribution for each sentiment seems to help.

+ve	low budget mostly no name actors. this is what a campy horror flick is supposed to be all about. these are the types of movies that kept me on the edge of my seat as a kid staying up too late to watch cable. if you liked the eighties horror scene this is the movie for you.
-ve	i and a friend rented this movie. we both found the movie soundtrack and production techniques to be lagging. the movie’s plot appeared to drag on throughout with little surprise in the ending. we both agreed that the movie could have been compressed into roughly an hour giving it more suspense and moving plot.

Table 5: Examples of reviews correctly classified by the cBLSTM LM classifier.

9 Conclusions

We have introduced a generative approach to sentiment analysis in which a novel contextual BLSTM (cBLSTM) LM is used as a sentiment classifier. Separate LM probability distributions are estimated for positive and negative sentiment from the training data. Then, probability scores from these LMs are utilized to classify test data. Results have been compared with a discriminative sentiment classification approach that uses a BLSTM-based binary classifier. We have observed that the generative cBLSTM LM approach significantly outperforms other approaches. Models have been evaluated on the IMDB large movie review dataset. The proposed models have achieved better results than the previously published results on the same dataset with the same partitioning. In addition, indicative comparisons have been made with the previously published results on the same dataset with different partitioning. Using model combi-

nation, we could achieve further performance improvement indicating that both the generative and discriminative approaches are complementary in solving the sentiment analysis problem. Moreover, we have introduced an efficient methodology based on perplexity calculation to partition the training data according to relative importance to the learning task. This partitioning methodology has been combined with a sub-sampling technique to efficiently train the neural networks on large data. As a future work, we plan to investigate deeper cBLSTM as well as hybrid recurrent and convolutional models. Another direction is to experiment with pre-trained word vectors.

Acknowledgments

The research leading to these results has received funding from the European Unions Horizon 2020 Programme through the Research and Innovation Action #645378 (ARIA-VALUSPA), the Innovation Action #644632 (MixedEmotions), as well as the German Federal Ministry of Education, Science, Research and Technology (BMBF) under grant agreement #16SV7213 (EmotAsS). We further thank the NVIDIA Corporation for their support of this research by Tesla K40 GPU donation.

References

- Tanel Alumäe and Mikko Kurimo. 2010. Efficient estimation of maximum entropy language models with N-gram features: an SRILM extension. In *Proc. Interspeech Conference of the International Speech Communication Association*, pages 1820–1823, Makuhari, Chiba, Japan, September.
- Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, and Stanely Chen. 2015. Bidirectional recurrent neural network language models for automatic speech recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5421–5425, Brisbane, Australia, April.
- Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179 – 190, March.
- Maximilian Bisani and Hermann Ney. 2004. Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 409 – 412, Montreal, Canada, May.
- George E. Dahl, Ryan Prescott Adams, and Hugo Larochelle. 2012. Training restricted boltzmann machines on word observations. In *Proc. International Conference on Machine Learning*, pages 679–686, Edinburgh, Scotland, UK, June.
- Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proc. International Conference on Web Search and Data Mining*, pages 231–240, Palo Alto, California, USA, February.
- Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proc. AAAI Conference on Artificial Intelligence*, pages 1537–1543, Québec, Québec, Canada, July.
- Li Dong, Furu Wei, Ke Xu, Shixia Liu, and Ming Zhou. 2016. Adaptive multi-compositionality for recursive neural network models. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 24(3):422–431.
- Volkmar Frinken, Alicia Fornés, Josep Lladós, and Jean-Marc Ogier, 2012. *Bidirectional Language Model for Handwriting Recognition*, pages 611–619. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 6645 – 6649, Vancouver, BC, Canada, May.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735 – 1780, November.
- Yi Hu, Ruzhan Lu, Yuquan Chen, and Jianyong Duan. 2007a. Using a generative model for sentiment analysis. *International Journal of Computational Linguistics & Chinese Language Processing*, 12(2):107–126, June.
- Yi Hu, Ruzhan Lu, Xuening Li, Yuquan Chen, and Jianyong Duan. 2007b. A language modeling approach to sentiment analysis. In *Proc. International Conference on Computational Science*, pages 1186–1193, Beijing, China, May.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 655–665, Baltimore, MD, USA, June.
- Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181 – 184, Detroit, Michigan, USA, May.

- Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. 2011. Recurrent neural network based language modeling in meeting recognition. In *Proc. Interspeech Conference of the International Speech Communication Association*, pages 2877 – 2880, Florence, Italy, August.
- Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proc. AAAI Conference on Artificial Intelligence*, pages 2267–2273, Austin, Texas, USA, January.
- Quoc V. Le and Tomáš Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. International Conference on Machine Learning*, pages 1188–1196, Beijing, China, June.
- Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proc. Joint Conference on Lexical and Computational Semantics*, pages 10–19, Denver, CO, USA, June.
- Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *Proc. AAAI Conference on Artificial Intelligence*, pages 1678–1684, Toronto, Ontario, Canada, July.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 142–150, Portland, Oregon, USA, June.
- Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan H. Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech Conference of the International Speech Communication Association*, pages 1045 – 1048, Makuhari, Chiba, Japan, September.
- Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 271 – 278, Barcelona, Spain, July.
- Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proc. Conference on Empirical Methods in NLP*, volume 10, pages 79–86, Philadelphia, PA, USA, July.
- Álvaro Peris and Francisco Casacuberta. 2015. A bidirectional recurrent neural language model for machine translation. *Procesamiento del Lenguaje Natural*, 55:109–116, September.
- Wenge Rong, Baolin Peng, Yuanxin Ouyang, Chao Li, and Zhang Xiong. 2014. Structural information aware deep semi-supervised recurrent neural network for sentiment analysis. *Frontiers of Computer Science*, 9(2):171–184.
- Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proc. International Conference on Computational Linguistics*, pages 69–78, Dublin, Ireland, August.
- Björn Schuller, Amr E. Mousa, and Vryniotis Vasileios. 2015. Sentiment analysis and opinion mining: On optimal parameters and performances. *WIREs Data Mining and Knowledge Discovery*, 5:255–263, September/October.
- Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. Conference on Empirical Methods in NLP*, pages 1631–1642, Seattle, WA, USA, October.
- Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proc. International Conference on Spoken Language Processing*, volume 2, pages 901 – 904, Denver, Colorado, USA, September.
- Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proc. Interspeech Conference of the International Speech Communication Association*, Portland, OR, USA, September.
- Maite Taboada, Julian Brooke, Milan Tofloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, June.
- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566, Beijing, China, July.
- Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 90 – 94, Jeju Island, Korea, July.
- Felix Weninger, Johannes Bergmann, and Björn Schuller. 2014. Introducing CURRENNT – the Munich open-source CUDA RecurREnt Neural Network Toolkit. *Journal of Machine Learning Research*, 15(99), October.
- Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2016. Achieving human parity in conversational speech recognition. Technical Report MSR-TR-2016-71, Microsoft Research, October.

Large-scale Opinion Relation Extraction with Distantly Supervised Neural Network

Changzhi Sun², Yuanbin Wu^{1,2}, Man Lan^{1,2},
Shiliang Sun^{1,2}, and Qi Zhang³

¹Shanghai Key Laboratory of Multidimensional Information Processing

²Department of Computer Science and Technology, East China Normal University

³School of Computer Science, Fudan University

{changzhisun}@stu.ecnu.edu.cn

{ybwu,mlan,slsun}@cs.ecnu.edu.cn

{qz}@fudan.edu.cn

Abstract

We investigate the task of open domain opinion relation extraction. Given a large number of unlabelled texts, we propose an efficient distantly supervised framework based on pattern matching and neural network classifiers. The patterns are designed to automatically generate training data, and the deep learning model is designed to capture various lexical and syntactic features. The result algorithm is fast and scalable on large-scale corpus. We test the system on the Amazon online review dataset, and show that the proposed model is able to achieve promising performances without any human annotations.

1 Introduction

Opinion mining systems aim to detect and extract opinion-related information from texts. With the help of natural language processing algorithms and large-scale user generated contents, researchers could take a closer look at how people express their opinions on various objects and topics. Such observations are important both for applications (e.g., recommendation and retrieval) and linguistic studies.

In this paper, we address the task of opinion relation extraction. The task tries to identify opinion expressions (words indicating sentiments, emotions and comments), opinion targets (objects of opinions) and their relations (what opinion on which target). The following are two examples.

1. The unit is [well designed] and [perfect reception].

2. The Passion of The Christ will [touch your heart].

Opinion relations in the two sentences are (“well designed”, “unit”), (“perfect reception”, “unit”), and (“touch your heart”, “The Passion of The Christ”). Extracting opinion bearing relations is usually the first step towards fine-grained analysis of opinion in texts, and plays an important role in other sentiment related applications (e.g., sentiment summarization). The goal of this paper is to extract opinion relations from open domain large-scale opinion bearing texts.

Previous works on fine-grained opinion information extraction have achieved notable success on many aspects: various domains were examined (Pontiki et al., 2015), different types of relations were studied (Ganapathibhotla and Liu, 2008; Narayanan et al., 2009; Wu et al., 2011), and both supervised and unsupervised (pattern-based) algorithms were applied. But we have observed some difficulties when trying to use existing methods. The pattern based methods (both lexical patterns and syntactic patterns) are simple, fast, and scalable on large-scale datasets. However, the robustness of patterns is usually questionable in practice. For example, syntactic patterns are sensitive to errors in parse trees, which are common in user generated contents. Lexical patterns either have limited coverage (e.g., fixed set of patterns (Riloff and Wiebe, 2003)), or hard-to-control noise (e.g., bootstrapping approaches (Qiu et al., 2011)). On the other hand, supervised models can achieve better performances than patterns on manually labeled datasets, but it is often difficult to obtain large number of annotations for the relation extraction task, and the trained models are

also limited to specified domains. Thus, we still need an algorithm to better combine the power of syntactic and lexical patterns, while reduce manual annotations.

Another problem is that many existing systems rely on general purpose opinion lexicons to select candidate relations. If an opinion expression is not recognized by the lexicon, the systems are unable to extract the related relations. As an example, one weakness of many existing lexicons is the lack of support on multiword expressions (e.g., “more than what I expected”, “honest to the book” and “adrenaline pumping”), which are common in opinion bearing texts. Another example is that some true expressions could be ignored either due to errors from POS taggers and syntactic parsers. Thus, to enlarge the coverage of opinion relation extraction, we need some method to better detect opinion expressions.

Regarding above problems, we make efforts to contribute to following aspects.

First, we propose a distantly supervised algorithm for open domain opinion relation extraction. The algorithm first applies domain independent patterns to get a set of opinion relations, then trains a classifier on them. We show that, although the relations from pattern matching are not as accurate as gold standard annotations, the distantly supervised classifier still improves performances. Our algorithm significantly outperforms the double propagation algorithm (Qiu et al., 2011), which is the state-of-the-art unsupervised opinion relation extraction system.

Second, we develop a neural network model to learn representations for lexical and syntactic contexts. The model uses bidirectional LSTMs to capture global information and convolutional neural networks to get local low dimensional feature embeddings. Comparing with other neural network models on relation extraction, we learn representations for different contexts explicitly, which is inspired by features in traditional relation classifiers. Empirical results show that the proposed model outperforms a strong logistic regression baseline, which uses handcrafted features as state-of-the-art supervised relation classifiers.

Third, we explore an unsupervised classifier to detect multiword opinion expressions. Given an expression, the classifier looks adjacent words and predicts whether it is an opinion expression. The new classifier helps us to discover opinion expres-

sions which are not in general purpose opinion lexicons, and benefits the relation extractor.

We aim to make all algorithms simple, fast and scalable for large-scale corpus. Our system is tested on Amazon review data which contains 15 different domains and 33 million reviews. The output database contains 72.5 million pairs of opinion relations. Extensive experiments have been conducted on various aspects of the algorithm, and the performances of the proposed unsupervised models are even competitive with previous supervised models.

2 Related Works

Opinion relation extraction is an important task for fine-grained sentiment analysis. If human annotations are provided (e.g. MPQA corpus (Deng and Wiebe, 2015)), we could formulate the task into a supervised relation extraction problem as (Kobayashi et al., 2007; Johansson and Moschitti, 2013). Two types of models have been applied: pipeline models which first extract candidates of opinion expressions and targets then identify correct relations (Wu et al., 2009; Li et al., 2010; Yang and Cardie, 2012), and joint models which extract opinion expressions, targets and relations using a unified joint model (Yang and Cardie, 2013; Yang and Cardie, 2014). One consideration of applying supervised methods is their dependencies on the domains and human annotations.

Semi-supervised and unsupervised models are also applied for extracting opinion relations. Approaches include rule-based bootstrapping (Qiu et al., 2011), graph propagation algorithms (Xu et al., 2013; Liu et al., 2014; Brody and Elhadad, 2010), integer programming (Lu et al., 2011), and probabilistic topic models (Titov and McDonald, 2008; Mukherjee and Liu, 2012).

Our model is inspired by previous distantly supervised algorithms (Snow et al., 2004; Mintz et al., 2009). They use relations from WordNet or knowledge bases as distant supervision. Since we don't have similar resources for opinion relation extraction, we use patterns to generate relations. Neural network classifiers are popular for relation extraction recently. Many of them focus on fully supervised settings, recurrent neural networks (RNN) and convolutional neural networks (CNN) (Vu et al., 2016; Zeng et al., 2015; Xu et al., 2015a; Xu et al., 2015b; Zhang and Wang, 2015), sequence models and tree models

are investigated (Li et al., 2015; dos Santos et al., 2015). One similar network structure to our model is proposed in (Miwa and Bansal, 2016). They jointly extract entities and relations using two LSTM models. Another recent work (Jebbara and Cimiano, 2016) uses stacked RNNs and CNNs for aspect and opinion detection. Different from models there, we will learn representations for different lexical and syntactic features explicitly. Our formulation follows the features in traditional relation classifiers, which helps to interpret the learned vectors.

A closely related task is aspect-based opinion mining (Zhao et al., 2010; Yu et al., 2011; Wang et al., 2015). Instead of locating the opinion expressions, aspect-based opinion mining directly analyzes polarities of different opinion targets. The targets are usually constrained to be some predefined set. Shared tasks (SemEval2014, SemEval2015) have been held on the task, and various systems are proposed and evaluated (Pontiki et al., 2014; Pontiki et al., 2015). Comparing with aspect-based opinion mining, we will extract opinion expressions which are more informative, and we won't constrain opinion target types which helps us to handle open domain texts.

3 The Approach

Given an input sentence $s = w_1, w_2, \dots, w_n$, where w_i is a word, the opinion relation extraction task outputs (O, T) pairs, where $O = w_i, w_{i+1}, \dots, w_j$ is an opinion expression, $T = w_k, w_{k+1}, \dots, w_l$ is an opinion target and the pair (O, T) is an opinion relation which asserts that opinion O is directed to target T ¹. Both O and T could be multiword expressions.

3.1 Patterns

Syntactic patterns have been shown to be effective for relation extraction. They are fast and can generalize well across domains, which are highly desirable for the open domain large-scale relation extraction task. However, despite of their advantages, two concerns are often raised: syntactic trees could be unreliable due to noise in texts and parsing errors, and the coverage of patterns is limited. To tackle the first problem, we deploy strong constraints on patterns in order to guarantee the quality of output. For the second problem, we en-

¹We assume O, T are non-overlapped, and their distance in s is less than a threshold (10 in all experiments).

large the coverage by using a distantly supervised classifier (Section 3.2) and an opinion expression classifier (Section 3.3)

Table 1 lists the patterns used in our system. Like (Qiu et al., 2011), patterns are based on the dependency tree of input sentences, which basically capture adjective-noun, verb-complement and adverb-verb relations. The notation $w_1 \xrightarrow{l} w_2$ denotes that there is a dependency relation between word w_1 and w_2 with dependency relation type l . For example, the pattern P1 is activated if w_1 is the parent of w_2 in the dependency tree, and the dependency type is `amod` or `dep`.

In order to overcome noise and errors in dependency trees, we constrain all patterns by predefined part-of-speech (POS) tag sets and a general purpose opinion lexicon L . For example, the pattern P1 only accepts nouns and adjectives as arguments, and the adjectives are required to be an opinion word in L .

We also design the patterns to be able to handle multiword opinion expressions and targets (about 30% of all annotated expressions). Two cases are considered here. First, when two words match a pattern, we expand them to the smallest phrases containing them. It helps to collect some local contexts of opinion relations. For example, in pattern P4, the matching words "case" and "choice" are expanded to "the case" and "an excellent choice". Second, a relation pair could be compiled to a new opinion expression, which may have relations with other opinion targets. For example, in pattern C2, ("perfectly", "fit") is a relation, and it can be compiled into "fit perfectly" which appears in a new relation ("fit perfectly", "the case"). The compiled expressions can bring more informative relations which are ignored in previous works.

As an alternative of pattern matching, we also investigate the bootstrapping setting like (Qiu et al., 2011). In this setting, the algorithm is allowed to add new words to the opinion lexicon, and use the updated lexicon for successive pattern matching. While the bootstrapping could discover new opinion words which are not in the original lexicon, we find that the errors caused by newly added words are hard to control, and the advantages of bootstrapping are suppressed by the noise as the corpus becomes large. We will show (in the experiment section) that the accuracy drops 30% comparing with the direct pattern matching.

Name	Pattern	Output	Example
P1	$w_1 \xrightarrow[\text{dep}]{\text{amod}} w_2$ $w_1.\text{pos} \in \text{NOUN}, w_2.\text{pos} \in \text{ADJ}$	$T = w_1.\text{np}$ $O = w_2$	It is a [cool] <u>case</u> . case $\xrightarrow{\text{amod}}$ cool
P2	$w_1 \xrightarrow[\text{xcomp}]{\text{acomp}} w_2$ $w_1.\text{pos} \in \text{VERB}, w_2.\text{pos} \in \text{ADJ}$	$T = w_1.\text{vp}$ $O = w_2$	The case <u>looks</u> [great]. looks $\xrightarrow{\text{xcomp}}$ great
P3	$w_1 \xrightarrow{\text{advmod}} w_2$ $w_1.\text{pos} \in \text{VERB}, w_2.\text{pos} \in \text{ADV}$	$T = w_1$ $O = w_2$	The cover <u>matches</u> [perfectly]. matches $\xrightarrow{\text{advmod}}$ perfectly
P4	$w_1 \xrightarrow{\text{nsbj}} w_2$ $w_1.\text{pos} \in \text{NOUN},$ $w_2.\text{pos} \in \text{NOUN or VERB or ADJ}$ has a coplua verb between w_1 and w_2	$T = w_1.\text{np}$ $(O = w_2.\text{np}$ $O = w_2.\text{vp}$ $O = w_2.\text{adjp})$	<u>This case</u> is [an excellent choice]. case $\xleftarrow{\text{nsbj}}$ choice
C1	$w_1 \xleftarrow{\text{conj}} (O', T')$ $O'.\text{pos.} \in \text{ADJ or ADV}$ $w_1.\text{pos} \in \text{ADJ or ADV}$	$T = T'$ $O = w_1$	The case <u>looks</u> [great] and very [cute]. cute $\xleftarrow{\text{conj}}$ (great, looks)
C2	$(O', T') \xrightarrow{\text{nsbj}} w_1$ $w_1.\text{pos} \in \text{NOUN}, T'.\text{pos} \in \text{VERB}$ $O'.\text{pos} \in \text{ADV}, T'$ and O' are adjacent	$T = w_1$ $O = T'O'$	<u>The case</u> [fits perfectly]. (perfectly, fits) $\xrightarrow{\text{nsbj}}$ the case

Table 1: Syntactic patterns in the system. We denote POS tag sets: NOUN = {NN, NNS}, VERB = {VB, VBD, VBN, VBP, VBZ}, ADJ = {JJ, JJR, JJS}, ADV = {RB, RBR, RBS}. $w_i.\text{pos}$ is the POS tag of w_i . $w_i.\text{np}$ ($w_i.\text{vp}$, $w_i.\text{adjp}$) is the smallest noun (verb, adjective) phrase containing w_i (return w_i if no such phrase exists). T, T' are opinion targets, O, O' are opinion expressions. “ $(O', T') \rightarrow$ ” and “ $\leftarrow (O', T')$ ” represent dependency relations on words O' and T' respectively.

3.2 Distant Supervision

Despite of the high precision, one well-known disadvantage of pattern-based methods is the low coverage. Consider the following example,

Ordered the k9ballistics Crate Pad and I
am [so pleased].

No pattern in Table 1 is applicable on relation (“so pleased”, “Ordered the k9ballistics Crate Pad”), although it could be inferred from the context. In fact, the two expressions are close in distance, and “Ordered the k9ballistics Crate Pad ” is the only possible object of “please” in the sentence. Many similar cases appear in online review corpus which downgrade the performance of patterns. To further explore those relations, we develop distantly supervised classifiers to integrate various lexical and syntactic contexts. Our experiments show that classifiers help to increase coverage of patterns by 20%.

Given a candidate relation $x = (O, T)$ in sentence s , the classifier outputs probability $p(y|x), y \in \{1, -1\}$ telling whether x is a valid opinion relation. Since manually labelled corpus are costly and difficult to obtain for open domains, we would prefer unsupervised classifiers. On the other side, the pattern matching can generate a set of opinion relations without any human annota-

tions. Although the relations are not completely correct, they are almost free to collect and easily amount to a large set. Thus, we can take the relations from the pattern matching as the distant supervision, and hope the broad coverage could overcome the noise.

Formally, we take all relations extracted by patterns as positive samples. For each positive sample (O, T) , we add a negative sample (O, T') for $T' \neq T$ (T' is NP, VP or ADJP). Similarly, we add negative samples (O', T) for all $O' \neq O$. At test time, we consider all VP and ADJP in s which contain at least one word in the general opinion lexicon L as candidate opinion expressions, all NP and VP as candidate opinion targets, and all possible pairs between them are candidate relations.

Our distantly supervised classifier is based on a neural network. Different from most previous deep learning models, the classifier learns representations for different lexical and syntactic contexts explicitly, which is inspired by features in traditional (non-neural-network-based) relation classifiers. We would observe from experiments that knowledge from previous feature engineering works can help neural network models to achieve better performances. Before explaining the model, we refresh some notations first. For $x = (O, T)$ in sentence s , where

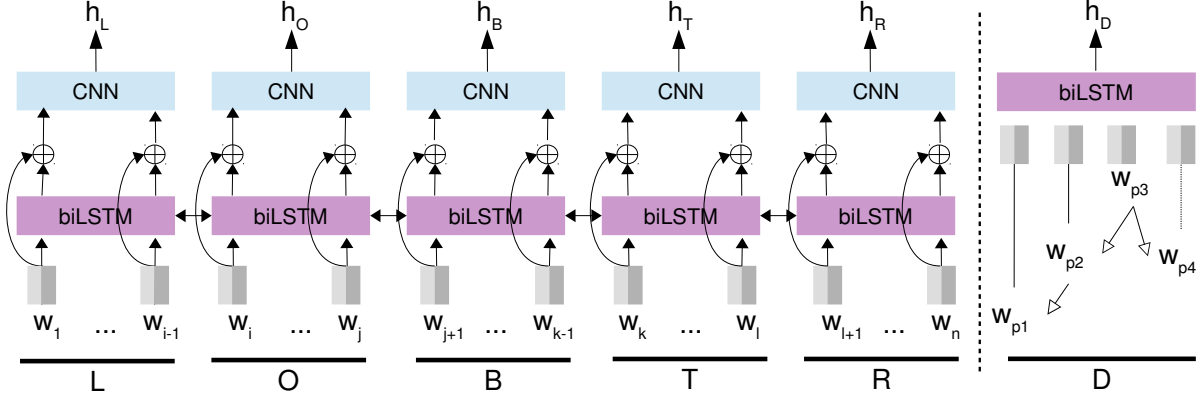


Figure 1: Representation learning of lexical and syntactic contexts. The left hand side is the five CNNs and the sentence level biLSTM for lexical contexts. For a word w , operator “ \oplus ” concatenates word/POS embeddings of w and the output vectors of the biLSTM on w . The right hand side is the biLSTM for syntactic contexts. w_{p1}, \dots, w_{p4} is the dependency path D , w_{p3} is the lowest common ancestor of head words of O and T .

$O = w_i, \dots, w_j, T = w_k, \dots, w_l$, denote other parts of s to be $L = w_1, \dots, w_{i-1}, B = w_{i+1}, \dots, w_{k-1}, R = w_{l+1}, \dots, w_n$, where L, R are the left and right contexts of x , B is the context between O and T ². Let $D = w_{p1}, \dots, w_{pm}$ be the path connecting head words of O and T in the dependency tree.

To capture lexical contexts of x , we use five convolutional neural networks (CNN) to learn representations for L, O, B, T, R . Take B as an example, the output $\mathbf{h}_B \in \mathbf{R}^d$ is the result of a single layer convolution of inputs with max-pooling. The input of the CNN includes word and POS tag embeddings of w_{i+1}, \dots, w_{k-1} . The five local CNN models can be independently trained before making the final predictions, however, it ignores global information of the sentence, and also the potential sharing of features among local models. In order to incorporate global structures, we build the five local models on top of a sentence level bidirectional long short term memory network (biLSTM). The recurrent structure and memory mechanism of biLSTM can propagate and share long distance features of s . We take outputs of memory cells as inputs of the CNN models (in addition to the embeddings). All representations $\mathbf{h}_L, \mathbf{h}_O, \mathbf{h}_B, \mathbf{h}_T, \mathbf{h}_R$ use the same network structure in our experiments. Finally, to make a prediction on y , we use a softmax function $p(y|x) = \frac{1}{Z} \exp\{\theta^T \Phi(x, y)\}$ on the weighted averaged fea-

²For simplicity, we assume O appears before T . In the implementation, an indicator dimension is set to identify whether O appears first.

ture vector $\Phi(x, y)$,

$$\Phi(x, y) = a_L \mathbf{h}_L + a_O \mathbf{h}_O + a_B \mathbf{h}_B + a_T \mathbf{h}_T + a_R \mathbf{h}_R,$$

where $a_L, a_O, a_B, a_T, a_R \in \mathbf{R}, \theta \in \mathbf{R}^d$ are parameters of the model.

We also try to incorporate dependency path D into the model as suggested by (Xu et al., 2015a; Xu et al., 2015b). We use a similar bidirectional LSTM like (Xu et al., 2015b), and concatenate the final outputs of the forward LSTM and the backward LSTM to get feature representation $\mathbf{h}_D \in \mathbf{R}^d$. The experiments show that, however, \mathbf{h}_D can not get further performance gains. We suspect that the errors from dependency parsing limited the contribution of this feature.

3.3 Opinion Expression Classifier

In above pattern matching and distant supervision algorithms, a candidate opinion expression is extracted if it contains at least one opinion word in the general purpose opinion lexicon L . Although the simple approach helps to handle multiword expressions, some expressions could also be ignored since they have no opinion words in L or their POS tags are wrongly assigned. In this section, we introduce our unsupervised opinion expression classifier, which predicts whether a phrase is an opinion expression based on its contexts.

Formally, for a candidate expression $O = w_i, \dots, w_j$ in sentence s , we use context words w_{i-c}, \dots, w_{i-1} and w_{j+1}, \dots, w_{j+c} as inputs of a CNN classifier (c is the context window size, and

we set it to 5 in all experiments). After the convolution layer, max-pooling and softmax (similar to the CNNs in Section 3.2), the classifier outputs probability $p(z|O)$ where $z \in \{-1, 1\}$ indicates whether O is a valid opinion expression. In order to get the training set, we rely on the lexicon L . Given the unlabelled corpus and $w \in L$, we consider each appearance of w in the corpus as a positive example, and other randomly chosen words as negative examples.

To apply the opinion expression classifier in the opinion relation classifier, we add expressions which have $p(z|O)$ greater than some threshold γ to the relation classifier.

4 Experiments

4.1 Configurations

We extract opinion relations on a subset of Amazon product review corpus provided by (McAuley et al., 2015), which contains 15 domains and 33 million reviews. The statistics of extracted relations are in Table 2.

For quantitative evaluation, we select four domains (Cell Phones, Movie and TV, Food, Pet Supplies) for detailed analyses. We manually label all correct opinion relations in 1000 sentences, and select 200 sentences as the development set, the rest 800 as the test set³. Furthermore, to compare with previous supervised methods, we also conduct experiments on USAGE corpus (Klinger and Cimiano, 2014) which annotates 4481 opinion relations for 8 products.

We use NLTK (Bird et al., 2009) for sentence splitting and word segmentation, Stanford parser⁴ for getting POS tags, phrase chunks and dependency trees, and scikit-learn toolkit (Pedregosa et al., 2011) and TensorFlow⁵ for machine learning algorithms. The general purpose opinion lexicon is from (Wilson et al., 2005).

4.2 Main Results

Table 4 shows results on four domains. The methods for comparison are:

- **Adjacent** is a simple baseline system from (Hu and Liu, 2004). It first identifies words in the general purpose opinion lexicon, then finds the nearest noun or verb phrase to them as their opinion targets.

³<https://github.com/AntNLP/OpinionRelationCorpus>

⁴<http://nlp.stanford.edu/software/lex-parser.shtml>

⁵<https://www.tensorflow.org/>

Domain	#Reviews	#Sents	#Relations
Cell Phones	3.4	19.1	6.7
Movie and TV	4.6	47.6	15.3
Food	1.3	7.6	2.5
Pet Supplies	1.2	8.0	2.4
Automotive	1.4	9.5	2.6
Digital Music	0.8	7.3	2.4
Beauty	2.2	6.4	3.8
Toys and Games	2.3	11.7	3.8
Instruments	0.5	13.5	1.4
Office Products	1.2	4.1	2.6
Patio	1.0	8.3	2.0
Baby	0.9	6.5	1.8
Clothing	5.7	29.1	10.2
Sports	3.3	20.2	7.1
Kindle	3.2	25.0	7.9
All	33	223.9	72.5

Table 2: Statistics of the opinion relation database. The relations are extracted by patterns in Table 1 and normalized by removing leading articles, pronouns and copulas of opinion expressions and targets. All numbers are in 10^6 .

- **Bootstrapping** reimplements the double propagation in (Qiu et al., 2011), which is the state-of-the-art unsupervised opinion relation extraction algorithm. It also uses a set of patterns, but adds new opinion words discovered by the patterns to the existing lexicon on the fly. The updated lexicon is then used in following bootstrapping iterations.
- **Pattern** is the pattern matching method in Section 3.1.
- **LR** is a logistic regression trained with the same distant supervision as Section 3.2. We use standard relation extraction features (Table 3), which are used in state-of-the-art supervised relation classifiers (Mintz et al., 2009)⁶.
- **NN** is our neural network model in Section 3.2. We set the dimension d of outputs (e.g., \mathbf{h}_B) be 128, the output dimension of biLSTM be 128, the dimension of word/POS tag embeddings be 300. We use three convolution kernels with window size 1, 2, 3, and initialize word embeddings with pre-trained word vectors from word2vec tools⁷. By default, we use the five CNNs on the sentence level biLSTM and not include dependency path \mathbf{h}_D and the opinion expression classifier (the configuration of “NN” equals “biLSTM+LOBTR” in Table 5). In order to build

⁶We select the features on the development set.

⁷<https://code.google.com/archive/p/word2vec/>

Lexical Features	
①	POS tag sequences of O and T .
②	The length of O and T .
③	The distance between O and T .
④	The word sequence between O and T in s .
⑤	POS tags of words between O and T in s .
⑥	Words, POS tags of $w_{i-1}, w_{i-2}, w_{j+1}, w_{j+2}$.
⑦	Words, POS tags of $w_{k-1}, w_{k-2}, w_{l+1}, w_{l+2}$.
⑧	Combined POS tags of O and T .
Syntactic Features	
⑨	Does a dependency relation exist between O and T .
⑩	The dependency path between O and T .
⑪	The length of the dependency path.

Table 3: Features in logistic regression.

the training set, we run the pattern matching on 6×10^4 unlabelled sentences.

- **NN+Pattern** stacks results of “Pattern” and “NN”.

We have several observations on Table 4. First, performances of “Adjacent” are poor, which means that we do need some advanced linguistic features for the task. Second, “Bootstrapping” underperforms “Pattern” in four domains. We examine the outputs of “Bootstrapping” and find that the newly added words bring a lot of noise into the opinion lexicon, which affect the accuracy negatively. Third, while “Pattern” has the highest precision in all systems, distantly supervised methods (“LR” and “NN”) help to improve recall and achieve better F1 values (except on the Pet domain). Hence, based on the distant supervision from patterns, classifiers cover more correct relations. Regarding the Pet domain, the precision of “Pattern” is low, so the number of errors in training set of “LR” and “NN” is large, and one could fail to learn reliable models. Fourth, the neural network model “NN” outperforms traditional classifier “LR” on all domains, which shows that learning feature representations has some advantages than handcrafting features on our experiment settings. Finally, simply stacking the results of “Pattern” and “NN” can improve the overall scores.

Next, we test our neural network model with various settings in Table 5. First, we compare models with different configurations on CNNs in row 1 to row 3. The setting “biLSTM+B” only uses the CNN corresponding to the words in B (i.e., the words between O and T); “biLSTM+OBT” uses three CNN on words in B, O, T ; “biLSTM+LOBTR” involves all five CNNs (equals to “NN” in Table 4). We see

that, in general, the performances (especially recalls) increase as we introduce more CNNs. However, the dependency path feature \mathbf{h}_D (“biLSTM+LOBTR+D” in row 4) won’t help to get further improvements. Second, in row 5, we drop the sentence level biLSTM and only use the five CNNs, and observe some loss on performances compared with row “biLSTM+LOBTR”. Hence, the long distance information provided by the biLSTM is also helpful. Third, we test the opinion expression classifier in the last two rows. Recall that γ is the threshold that controls the output of the classifier. The result shows that when $\gamma = 0.8$, new opinion expressions added by the classifier can improve the scores, but when $\gamma = 0.5$, the noise can overwhelm the gains. We further show the precision-recall curves of “NN” and “LR” in the case of $\gamma = 0.8$ in Figure 2. Some interesting opinion expressions added by the classifier are “not even enough”, “became extremely hot”, “*just* enough”, “arrived damaged”, “looks cool n cute”, which shows that the classifier could both discover opinion expressions without words in general purpose lexicons, and have some tolerance to noise.

4.3 Results on USAGE Corpus

In order to compare with fully supervised methods, we evaluate our models on USAGE corpus in Table 6. To build the distantly supervised models, we use untagged reviews which are about the 8 products of USAGE. The baseline systems are (Klinger and Cimiano, 2014) and (Jebbara and Cimiano, 2016), which are state-of-the-art systems on the dataset.

Results show that, with the same setting (“gold”) ⁸, our fully unsupervised models achieve competitive precision scores against previous fully supervised methods. By examining the outputs, one reason for the performance gaps on recall may be the differences of annotation guidelines between USAGE and our dataset. For example, we don’t annotate pronouns as opinion targets while USAGE does (e.g., (“love”, “it”) is a proper annotation in USAGE). We can also observe that distant supervisions provide notable performances gains than direct pattern matching.

⁸Both baselines don’t report end-to-end performances. As a reference, in (Jebbara and Cimiano, 2016), the F1 of opinion expression and target extraction are 50% and 67%.

System	Phone			Movie			Food			Pet		
	P	R	F	P	R	F	P	R	F	P	R	F
Adjacent	38.6	65.7	48.6	30.0	58.8	39.7	31.4	46.5	37.5	28.4	62.2	39.0
Bootstrapping	44.0	62.9	51.8	26.9	49.3	34.8	43.6	54.0	48.2	33.6	57.7	42.5
Pattern	69.4*	64.4	61.1	62.2*	42.4	50.4	76.0*	41.9	54.1	59.9*	51.3	55.3*
LR	60.1	64.7	62.4	55.6	57.0	55.3	65.5	49.2*	56.2	47.6	59.3*	52.8
NN	63.4	67.9*	65.6*	56.8	58.2*	57.5*	70.5	47.7	56.9*	51.4	58.0	54.5
NN+Pattern	64.4	70.5	67.3	58.2	59.9	59.1	68.4	50.8	58.3	54.9	58.2	56.5

Table 4: Comparison with different baseline systems. The dark entries are the highest scores among all systems, and “*” indicates the highest scores excluding “NN+Pattern”.

System	Phone			Movie			Food			Pet		
	P	R	F	P	R	F	P	R	F	P	R	F
biLSTM +B	59.8	69.5	64.3	54.7	59.1	56.8	64.8	48.5	55.5	47.4	57.2	51.8
biLSTM +OBT	63.4	66.7	65.3	59.8	58.2	58.9	68.6	46.8	55.6	56.2	56.9	56.5
biLSTM +LOBTR	63.4	67.9	65.6	56.8	58.2	57.5	66.5	47.7	55.5	51.4	58.0	54.5
biLSTM +LOBTR+D	65.5	61.3	63.4	59.5	55.8	57.6	69.1	46.8	55.8	54.9	57.7	56.3
LOBTR	64.0	65.3	64.7	57.5	56.7	57.1	70.6	43.1	53.5	54.3	57.7	55.9
$\gamma = 0.5$	64.2	64.7	64.5	56.2	57.9	57.0	65.5	45.5	53.7	61.6	54.3	57.7
$\gamma = 0.8$	65.6	66.1	65.9	61.1	58.2	59.6	67.1	48.2	56.1	58.6	50.8	54.4

Table 5: Different settings of the proposed model.

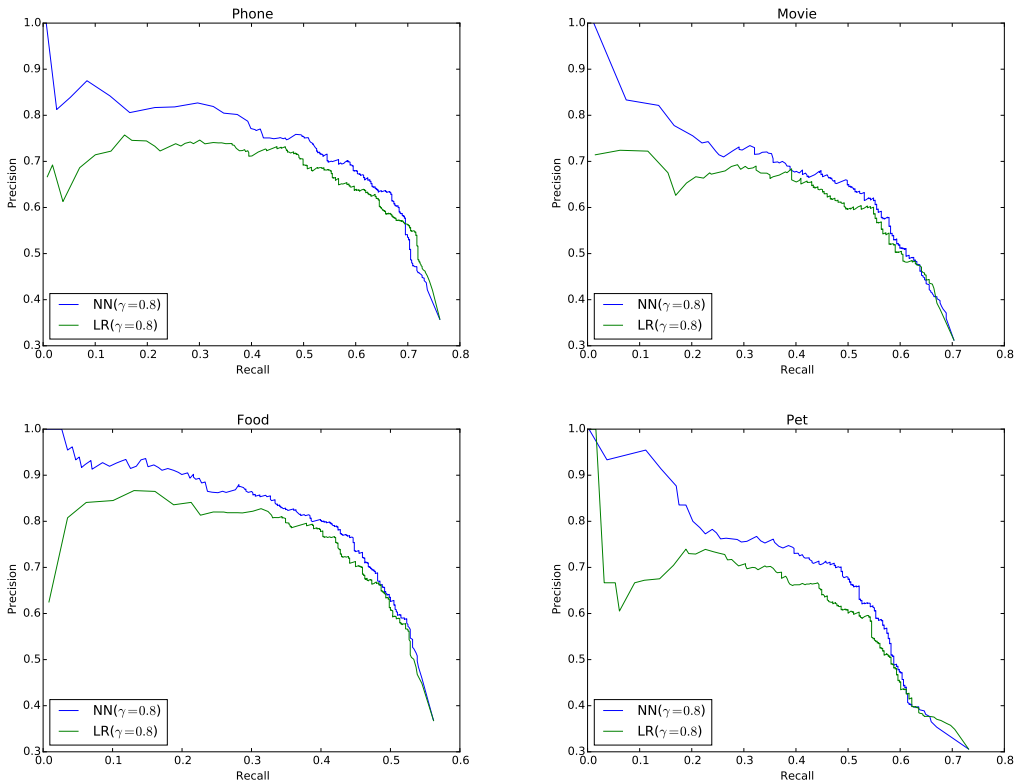


Figure 2: Precision recall curves for $\gamma = 0.8$.

4.4 Error Analysis

Finally, we do some error analyses on the extracted relations. Taking movie domain as example, we find that for movie reviews, the opinions

from reviewers are mixed with plot and characters of movies, which makes it difficult to distinguish opinions and background topics with our simple opinion relation definition. For example,

Systems	P	R	F
Klinger et al. (2014)	-	-	65.0
Jebbara et al. (2016)	87.0	75.0	81.0
Pattern	51.4	20.7	29.5
LR (end-to-end)	49.5	27.8	35.6
NN (end-to-end)	43.3	40.1	41.6
LR (gold)	89.1	47.9	62.3
NN (gold)	81.4	62.8	70.9

Table 6: Results on USAGE corpus. First two rows are state-of-the-art systems on the dataset. Both of them assume gold annotations on opinion expressions and targets have been given. We report results with identical settings (“gold”), and also “end-to-end” results in which no gold annotations are provided.

in “Also said repeatedly how Tojo was [loyal to Emperor Hirohito]”, the word “loyal” indicates a wrong opinion relation since it’s a description of the story. A true comment from the reviewer is behind the word “repeatedly”, which is hard to be expressed with our opinion relations. We plan to introduce both more background knowledge and more powerful relation types in future work.

5 Conclusion

We investigate the task of large-scale opinion relation extraction. Our algorithm first uses syntactic patterns to get a set of opinion relations, then builds a neural network classifier based on these relations. We also develop an opinion expression classifier to better extract opinion words. Extensive experiments on Amazon review data show the effectiveness of the proposed methods.

6 Acknowledgements

The authors wish to thank the reviewers for their helpful comments and suggestions. This research is supported by NSFC (61402175, 61473092) and STCSM (15ZR1410700, 14DZ2260800). Yuanbin Wu is supported by Microsoft Research Asia Collaborative Research Program. The corresponding authors are Yuanbin Wu, Man Lan and Shiliang Sun.

References

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

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812, Los Angeles, California, June. Association for Computational Linguistics.

Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1323–1328, Denver, Colorado, May–June. Association for Computational Linguistics.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China, July. Association for Computational Linguistics.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 241–248.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of SIGKDD*, pages 168–177.

Soufian Jebbara and Philipp Cimiano. 2016. Aspect-based relational sentiment analysis using a stacked neural network architecture. In *ECAI*, pages 1123–1131.

Richard Johansson and Alessandro Moschitti. 2013. Relational features in fine-grained opinion analysis. *Computational Linguistics*, 39(3):473–509.

Roman Klinger and Philipp Cimiano. 2014. The US-AGE review corpus for fine grained multi lingual opinion analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 2211–2218.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1065–1074, Prague, Czech Republic, June. Association for Computational Linguistics.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Yingju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 653–661.

- Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proc. of EMNLP*, pages 2304–2314.
- Kang Liu, Liheng Xu, and Jun Zhao. 2014. Extracting opinion targets and opinion words from online reviews with graph co-ranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 314–324, Baltimore, Maryland, June. Association for Computational Linguistics.
- Yue Lu, Malú Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proc. of WWW*, pages 347–356.
- Julian J. McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 785–794.
- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August. Association for Computational Linguistics.
- Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany, August. Association for Computational Linguistics.
- Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 339–348, Jeju Island, Korea, July. Association for Computational Linguistics.
- Ramanathan Narayanan, Bing Liu, and Alok Choudhary. 2009. Sentiment analysis of conditional sentences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 180–189, Singapore, August. Association for Computational Linguistics.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830.
- Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495.
- Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.
- Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 105–112.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, pages 1297–1304.
- Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, pages 308–316, Columbus, Ohio, June. Association for Computational Linguistics.
- Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, San Diego, California, June. Association for Computational Linguistics.
- Linlin Wang, Kang Liu, Zhu Cao, Jun Zhao, and Gerard de Melo. 2015. Sentiment-aspect extraction based on restricted boltzmann machines. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 616–625, Beijing, China, July. Association for Computational Linguistics.
- Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.
- Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on*

- Empirical Methods in Natural Language Processing*, pages 1533–1541, Singapore, August. Association for Computational Linguistics.
- Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2011. Structural opinion mining for graph-based sentiment representation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1332–1341, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Liheng Xu, Kang Liu, Siwei Lai, Yubo Chen, and Jun Zhao. 2013. Mining opinion words and opinion targets in a two-stage framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1764–1773, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540, Lisbon, Portugal, September. Association for Computational Linguistics.
- Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal, September. Association for Computational Linguistics.
- Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345, Jeju Island, Korea, July. Association for Computational Linguistics.
- Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Bishan Yang and Claire Cardie. 2014. Joint modeling of opinion expression extraction and attribute classification. *Transactions of the Association of Computational Linguistics*, 2:505–516.
- Jianxing Yu, Zheng-Jun Zha, Meng Wang, Kai Wang, and Tat-Seng Chua. 2011. Domain-assisted product aspect hierarchy generation: Towards hierarchical organization of unstructured consumer reviews. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 140–150, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal, September. Association for Computational Linguistics.
- Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *CoRR*.
- Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65, Cambridge, MA, October. Association for Computational Linguistics.

Decoding with Finite-State Transducers on GPUs

Arturo Argueta and David Chiang

Department of Computer Science and Engineering

University of Notre Dame

aargueta,dchiang@nd.edu

Abstract

Weighted finite automata and transducers (including hidden Markov models and conditional random fields) are widely used in natural language processing (NLP) to perform tasks such as morphological analysis, part-of-speech tagging, chunking, named entity recognition, speech recognition, and others. Parallelizing finite state algorithms on graphics processing units (GPUs) would benefit many areas of NLP. Although researchers have implemented GPU versions of basic graph algorithms, limited previous work, to our knowledge, has been done on GPU algorithms for weighted finite automata. We introduce a GPU implementation of the Viterbi and forward-backward algorithm, achieving decoding speedups of up to 5.2x over our serial implementation running on different computer architectures and 6093x over OpenFST.

1 Introduction

Weighted finite automata (Mohri, 2009), including hidden Markov models and conditional random fields (Lafferty et al., 2001), are used to solve a wide range of natural language processing (NLP) problems, including phonology and morphology, part-of-speech tagging, chunking, named entity recognition, and others. Even models for speech recognition and phrase-based translation can be thought of as extensions of finite automata (Mohri et al., 2002; Kumar et al., 2005).

Although the use of graphics processing units (GPUs) is now *de rigueur* in applications of neural networks and made easy through toolkits like Theano (Theano Development Team, 2016), there has been little previous work, to our knowledge,

on acceleration of weighted finite-state computations on GPUs (Narasiman et al., 2011; Li et al., 2014; Peng et al., 2016; Chong et al., 2009). In this paper, we consider the operations that are most likely to have high speed requirements: decoding using the Viterbi algorithm, and training using the forward-backward algorithm. We present an implementation of the Viterbi and forward-backward algorithms for CUDA GPUs. We release it as open-source software, with the hope of expanding in the future to a toolkit including other operations like composition.

Most previous work on parallel processing of finite automata (Ladner and Fischer, 1980; Hillis and Steele, 1986; Mytkowicz et al., 2014) uses dense representations of finite automata, which is only appropriate if the automata are not too sparse (that is, most states can transition to most other states). But the automata used for natural language tend to be extremely large and sparse. In addition, the more recent work in this line assumes deterministic automata, but automata that model natural language ambiguity are generally nondeterministic.

Previous work has been done on accelerating particular NLP tasks on GPUs: in machine translation, phrase-pair retrieval (He et al., 2013) and language model querying (Bogoychev and Lopez, 2016); parsing (Hall et al., 2014; Canny et al., 2013); and speech recognition (Kim et al., 2012). Our aim here is for a more general-purpose collection of algorithms for finite automata.

Our work uses concepts from the work of Merrill et al. (2012), who show that GPUs can be used to accelerate breadth-first search in sparse graphs. Our approach is simple, but well-suited to the large, sparse automata that are often found in NLP applications. We show that it achieves a speedup of a factor of 5.2 on a GPU relative to a serial algorithm, and 6093 relative to OpenFST.

2 Graphics Processing Units

GPUs became known for their ability to render high quality images faster than conventional multi-core CPUs. Current off-the-shelf CPUs contain 8–16 cores while GPUs contain 1500–2500 simple CUDA cores built into the card. General Purpose GPUs (GPGPU) contain cores able to execute calculations that are not constrained to image processing. GPGPUs are now widely used across scientific domains to enhance the performance of diverse applications.

2.1 Architecture

CUDA cores (also known as scalar processors) are grouped into different Streaming Multiprocessors (SM) on the graphics card. The number of cores per SM varies depending on the GPU’s micro-architecture, ranging from 8 cores per SM (Tesla) up to 192 (Kepler). The overall number of SM on the chip varies, and it can range from 15 (Kepler) up to 24 (Maxwell). Streaming Multiprocessors are composed of the following components:

- **Special Function units (SFU)** These allow computations of functions such as sine, cosine, etc.
- **Shared Memory and L1 Cache** The size of the memory varies on the GPU model.
- **Warp Schedulers** assigns threads in an SM to be executed in a specific warp.

To execute a workload on the GPU, a kernel must be launched with a specified grid structure. The kernel must specify the number of threads to run on a block and the number of blocks in a grid before being executed on the device. The maximum number of threads per block and blocks per grid can vary depending on the GPU device. If the kernel is successfully launched, each block in the grid will get assigned to a SM. Each SM will execute 32 threads at a time (also called a warp) in its assigned block. If the number of threads in a block is not divisible by 32, the kernel will not launch on the device. Each SM contains a warp scheduler in charge of choosing the warps in a block to be executed in parallel. When the amount of blocks in a grid surpasses the amount of SM on the device, the SMs will execute a subset of blocks in parallel.

K40 Specs	
Global Memory	11520 MB
L2 cache size	1.57 MB
Shared memory per block	0.049 MB
Multiprocessors	15
Cores per MP	192
Registers per block	65536

Table 1: Device properties of a K40c GPU

The memory hierarchy on the device is laid out to maximize the data throughput. Table 1 shows the amount of cores available for execution as well as the amount of memory available on a Kepler based GPU. Registers are the fastest type of memory on the device, and this memory is private to each thread running on a block. Shared memory is the second fastest, and is shared by all threads running in the same block. The next type of memory is the L2 cache, which is shared among all streaming multiprocessors. The slowest and largest type of memory is global memory. Directly reading and writing to global memory affects performance significantly. Efficient memory management (reading and writing to and from contiguous addresses in memory) is important to fully utilize the memory hierarchy and increase performance.

2.2 Optimizations

Different factors such as number of threads in a block or coalesced memory accesses affect the performance on the GPU. In this section, we will cover the methods and modifications we used to improve the performance of our parallel implementations.

The optimal number of threads per block depends on the device configuration. The number of multiprocessors and cores per multiprocessor must be considered before launching a CUDA kernel on the device. Table 1 shows the number of streaming multiprocessors and the number of cores per multiprocessor on a K40 GPU. Multiple blocks in a kernel grid can get scheduled to be executed on a single streaming multiprocessor if the number of blocks in a grid exceeds the number of streaming multiprocessors. Each streaming multiprocessor will only execute one warp in a block in parallel during execution, and that is why choosing an appropriate number of blocks is important. For example, if two blocks get assigned to a multiprocessor and each block contains 192 threads, the

multiprocessor must execute 12 warps total where 1 warp gets executes at a time in parallel.

In our implementations, we take the following approach. The number of cores per multiprocessor is considered first to configure the block size. The block size is set to contain the same number of threads as the number of cores per multiprocessor of the graphics card used. If the number of threads needed to perform a computation is not divisible by the amount of cores per multiprocessor, the number of threads is rounded up to the closest dividend. Once the block size and number of threads are selected, the number of blocks is chosen by dividing the total number of threads by the block size.

Coalesced memory accesses are essential to maximize the use of resources running on the GPU. When data is requested by a warp executing on a streaming multiprocessor, a block from global memory will be accessed and allocated in shared memory. It is crucial to coalesce memory accesses so the number of blocks of global memory requested and the global memory access times decrease. This can be achieved by making all threads in a warp access contiguous spaces in memory. A similar speedup can be achieved if each thread in a block allocates all the data required from global memory into a compact data structure allocated in shared memory (size of the shared memory varies across devices). Section 4 describes the data structure used to coalesce memory reads. For each input symbol w_t the source states of all possible transitions can be read in a coalesced form and stored in shared memory allowing faster execution times.

Using special function units on the device can inhibit the performance of a program running on the GPU. Performance is affected because the number of SFU is lower than the amount of regular cores (e.g. The GK104 Kepler architecture contains 1536 regular cores and 256 special function units total). Also, the cycle penalty for using SFU rather than CUDA cores is higher than the penalty for regular cores on the device. For this work, the amount of instructions that use a specific SFU are kept to a minimum to obtain a higher speedup. By combining the mentioned techniques in this section, an application can significantly increase its performance.

3 Weighted Finite Automata

In this section, we review weighted finite automata, using a matrix formulation. A *weighted finite automaton* is a tuple $M = (Q, \Sigma, s, F, \delta)$, where

- Q is a finite set of states.
- Σ is a finite input alphabet.
- $s \in \mathbb{R}^Q$ is a one-hot vector: if M can start in state q , then $s[q] = 1$; otherwise, $s[q] = 0$.
- $f \in \mathbb{R}^Q$ is a vector of final weights: if M can accept in state q , then $f[q] > 0$ is the weight incurred; otherwise, $f[q] = 0$.
- $\delta : \Sigma \rightarrow \mathbb{R}^{Q \times Q}$ is the transition function: if M is in state q and the next input symbol is a , then $\delta[a][q, q']$ is the weight of going to state q' .

Note that we currently do not allow transitions on empty strings or epsilon transitions. This definition can easily be extended to weighted finite transducers by augmenting the transitions with output symbols. See Figure 1 for an example FST.

Using this notation, the total weight of a string $w = w_1 \cdots w_n$ can be written succinctly as:

$$\text{weight}(w) = s^\top \left(\prod_{t=1}^n \delta[w_t] \right) f. \quad (1)$$

Matrix multiplication is defined in terms of multiplication and addition of weights. It is common to redefine weights and their multiplication/addition to make the computation of (1) yield various useful values. When this is done, multiplication is often written as \otimes and addition as \oplus . If we define $p_1 \otimes p_2 = p_1 p_2$ and $p_1 \oplus p_2 = p_1 + p_2$, then equation (1) gives the total weight of the string.

Or, we can make Equation (1) obtain the *maximum* weight path as follows. The weight of a transition is (p, k) , where p is the probability of the transition and k is (a representation of) the transition itself. Then

$$(p_1, k_1) \otimes (p_2, k_2) \equiv (p_1 p_2, k_1 k_2)$$

$$(p_1, k_1) \oplus (p_2, k_2) \equiv \begin{cases} (p_1, k_1) & \text{if } p_1 > p_2 \\ (p_2, k_2) & \text{otherwise.} \end{cases}$$

The Viterbi algorithm simply computes Equation (1) under the above definition of weights.

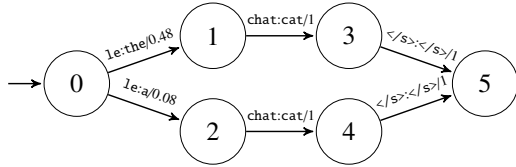


Figure 1: Example of a FST that translates the french string `le chat` to English.

4 Serial Algorithm

Applications of finite automata use a variety of algorithms, but the most common are the Viterbi, forward, and backward algorithms. Several of these automata algorithms are related to one another and used for learning and inference. Speeding up these algorithms will allow faster training and development of large scale machine learning systems.

The forward and backward algorithms are used to compute weights (Eq. 1), in left-to-right (Reading an input utterance from left to right) and right-to-left order, respectively. Their intermediate values are used to compute expected counts during training by expectation-maximization (Eisner, 2002). They can be computed by Algorithm 2.

Algorithm 1 is one way of computing Viterbi using Equation (1). It is a straightforward algorithm, but the data structures require a brief explanation.

Throughout this paper, we use zero-based indexing for arrays. Let $m = |\Sigma|$, and number the input symbols in Σ consecutively $0, \dots, m - 1$. Then we can think of δ as a three-dimensional array. In general, this array is very sparse. We store it using a combination of compressed sparse row (CSR) format and coordinate (COO) format, as shown in Figure 2 where:

- z is the number of transitions with nonzero weight
- R is an array of length $(m + 1)$ containing offsets into the arrays S, T, O , and P . if $a \in \Sigma$, the transitions on input a can be found at positions $R[a], \dots, R[a + 1] - 1$ (i.e. to access all transitions $\delta[a]$). Note that $R[m] = z$
- S contains the source states for each transition $0 \leq k < z \in \delta[a]$
- T contains target states for transitions $0 \leq k < z \in \delta[a]$

	le	chat	</s>	</s>	
R	0	2	4	6	
S	0	0	1	2	3
T	1	2	3	4	5
O	the	a	cat	cat	</s> </s>
P	0.48	0.08	1	1	1

Figure 2: CSR/COO representation of FST in Figure 1.

- O contains the output symbols for transitions from state $S[k]$ to state $T[k]$
- P contains the probabilities for transitions from state $S[k]$ to state $T[k]$

The vector f of final weights is stored as a sparse vector: for each k , $S_f[k]$ is a final state with weight $P_f[k]$.

Algorithm 1 Serial Viterbi algorithm (using CSR/COO representation).

```

1: for  $q \in Q$  do
2:    $\alpha[0][q] = 0$ 
3:  $\alpha[0][s] = 1$ 
4: for  $t = 1, \dots, n$  do
5:    $a \leftarrow w_t$ 
6:   for  $k = R[a], \dots, R[a + 1] - 1$  do
7:      $p \leftarrow \alpha[t - 1][S[k]] \otimes P[k]$ 
8:      $\alpha[t][T[k]] \leftarrow \alpha[t][T[k]] \oplus p$ 
9: return  $\bigoplus_k \alpha[n][S_f[k]] \otimes P_f[k]$ 

```

If the transition matrices $\delta[a]$ are stored in compressed sparse row (CSR) format, which enables efficient traversal of a matrix in row-major order, then these algorithms can be written out as Algorithm 2 for the forward-backward algorithm and 1 for Viterbi. (Using compressed sparse columns (CSC) format, the loop over q' would be outside the loop over q , which is perhaps the more common way to implement these algorithms.)

5 Parallel Algorithm

Our parallel implementation is based on Algorithm 1 for Viterbi and Algorithm 2 for forward-backward, but parallelizes the loop over t , that is, over the transitions on symbol w_t . The transitions

Algorithm 2 Forward-Backward algorithm (row-major).

```

1: forward[0][s] ← 1      ▶ Begin forward pass
2: for t = 0, ..., n - 1 do
3:   for q ∈ Q do
4:     for q' ∈ Q such that δ[wt+1][q, q'] > 0 do
5:       p = forward[t][q]δ[wt+1][q, q']
6:       forward[t + 1][q'] += p
7:   for q ∈ Q do      ▶ backward pass
8:     backward[n][q] = f[q]
9:   for t = n - 1, ..., 0 do
10:    for q ∈ Q do
11:     for q' ∈ Q such that δ[wt+1][q, q'] > 0 do
12:      p = δ[wt+1][q, q']backward[t][q']
13:      backward[t][q] += p
14: Z = ∑q∈Q forward[n][q]f[q]
15: for t = 0, ..., n - 1 do
16:   for q, q' ∈ Q do
17:    α = forward[t][q]      ▶ Expected counts
18:    β = backward[t + 1][q']
19:    count[q, q'] += α × δ[w][q, q'] × β/Z

```

are stored in CSR/COO format as described above for Algorithm 1. The S , T , and P arrays are stored on the GPU in global memory; the R and O arrays are kept on the host. For each input symbol a , the transitions on S and T are sorted first by source state and then by target state; this improves memory locality slightly. For the forward-backward algorithm, sorting by target improves the performance for the backward pass since the input is read from right to left.

For each input symbol w_t , one thread is launched per transition, that is, for each nonzero entry of the transition matrix $\delta[w_t]$. Equivalently, one thread is launched for each transition k such that $R[w_t] \leq k < R[w_t + 1]$, for a total of $R[w_t + 1] - R[w_t]$ threads. Each thread looks up $q = S[k]$, $q' = T[k]$ and computes its corresponding operation.

For example, in Figure 2, input word “le” has index 0; since $R[0] = 0$ and $R[1] = 2$, two threads are launched, one for $k = 0$ (that is, $0 \xrightarrow{\text{le:the}/0.48} 1$) and one for $k = 1$ (that is, $0 \xrightarrow{\text{le:a}/0.08} 2$).

5.1 Viterbi

At the time of computing a transition $\delta[w_t][q, q']$, if the probability (at line 8 in Algorithm 1) is higher than $\alpha[t][q']$, we store the probability in $\alpha[t][q']$. Because this update potentially involves

concurrent reads and writes at the same memory location, we use an atomic max operation (defined as `atomicMax` on the NVIDIA toolkit). However, `atomicMax` is not defined for floating-point values. Additionally, this update needs to store a back-pointer (k) that will be used afterwards to reconstruct the highest-probability path. The problem is that the `atomicMax` provided by NVIDIA can only update a single value atomically.

We solve both problems with a trick: pack the Viterbi probability and the back-pointer into a single 64-bit integer, with the probability in the higher 32 bits and the back-pointer in the lower 32 bits. In IEEE 754 format, the mapping between nonnegative real numbers and their bit representations (viewed as integers) is order-preserving, so a max operation on this packed representation updates both the probability and the back-pointer simultaneously.

The reconstruction of the Viterbi path is not parallelizable, but is done on the GPU to avoid copying α back to the host avoiding a slowdown. This generates a sequence of transition indices, which is moved back to the host. There, the output symbols can be looked up in array O .

5.2 Forward-Backward

The forward and backward algorithms 2 are similar to the Viterbi algorithm, but do not need to keep back-pointers. In the forward algorithm, when a transition $\delta[w_t][q, q']$ is processed, we update the sum of probabilities reaching state q' in $\text{forward}[t + 1][q']$. Likewise, in the backward algorithm, we update the sum of probabilities starting from q in $\text{backward}[t][q]$. Both passes require atomic addition operations, but because we use log-probabilities to avoid underflow, the atomic addition must be implemented as:

$$\log(\exp a + \exp b) = b + \log 1p(\exp(a - b)), \quad (2)$$

assuming $a \leq b$ and where $\log 1p(x) = \log(1 + x)$, a common function in math libraries which is more numerically stable for small x .

We implemented an atomic version of this log-add-exp operation. The two transcendentals are expensive, but CUDA’s fast math option (`-use_fast_math`) speeds them up somewhat by sacrificing some accuracy.

6 Other Approaches

6.1 Parallel prefix sum

We have already mentioned a line of work begun by (Ladner and Fischer, 1980) for unweighted, nondeterministic finite automata, and continued by (Hillis and Steele, 1986) and (Mytkowicz et al., 2014) for unweighted, deterministic finite automata. These approaches use *parallel prefix sum* to compute the weight (1), multiplying each adjacent pair of matrices in parallel and repeating until all the matrices have been multiplied together.

This approach could be combined with ours; we leave this for future work. A possible issue is that matrix-vector products are replaced with slower matrix-matrix products. Another is that prefix sum might not be applicable in a more general setting – for example, if a FST is composed with an input lattice rather than an input string.

6.2 Matrix libraries

The formulation of the Viterbi and forward-backward algorithms as a sequence of matrix multiplications suggests two possible easy implementation strategies. First, if transition matrices are stored as dense matrices, then the forward algorithm becomes identical to forward propagation through a rudimentary recurrent neural network. Thus, a neural network toolkit could be used to carry out this computation on a GPU. However, in practice, because our transition matrices are sparse, this approach will probably be inefficient.

Second, off-the-shelf libraries exist for sparse matrix/vector operations, like cuSPARSE.¹ However, such libraries do not allow redefinition of the addition and multiplication operations, making it difficult to implement the Viterbi algorithm or use log-probabilities. Also, parallelization of sparse matrix/vector operations depends heavily on the sparsity pattern (Bell and Garland, 2008), so that an off-the-shelf library may not provide the best solution for finite-state models of language. We test this approach below and find it to be several times slower than a non-GPU implementation.

7 Experiment

7.1 Setup

To test our algorithm, we constructed a FST for rudimentary French-to-English translation. We trained different unsmoothed bigram language

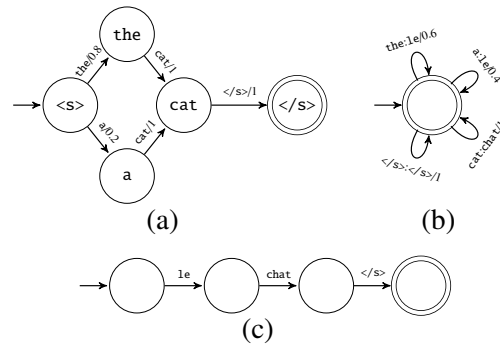


Figure 3: Example automata/transducers for (a) language model (b) translation model (c) input sentence. These three composed together form the transducer in Figure 1.

models on 1k/10k/100k/150k lines of French-English parallel data from the Europarl corpus and converted it into a finite automaton (see Figure 3a for a toy example).

GIZA++ was used to word-align the same data and generate word-translation tables $P(f | e)$ from the word alignments, as in lexical weighting (Koehn et al., 2003). We converted this table into a single-state FST (Figure 3b). The language model automaton and the translation table transducer were intersected to create a transducer similar to the one in Figure 1.

For more details about the transducers (number of nodes, edges, and percentage of non-zero elements on the transducer) see Table 4.

We tested on a subset of 100 sentences from the French corpus with lengths of up to 80 words. For each experimental setting, we ran on this set 1000 times and report the total time. Our experiments were run on three different systems: (1) a system with an Intel Core i7-4790 8-core CPU and an NVIDIA Tesla K40c GPU, (2) a system with an Intel Xeon E5 16-core CPU and an NVIDIA Titan X GPU, and (3) a system with an Intel Xeon E5 24-core CPU and an NVIDIA Tesla P100 GPU.

7.2 Baselines

We compared against the following baselines:

Carmel is an FST toolkit developed at USC/ISI.²

OpenFST is a FST toolkit developed by Google as an open-source successor of the AT&T Finite State Machine library (Allauzen et al., 2007). For compatibility, our implementations read the OpenFST/AT&T text file format.

¹<http://docs.nvidia.com/cuda/cuspars/>

²<https://github.com/graehl/carmel>

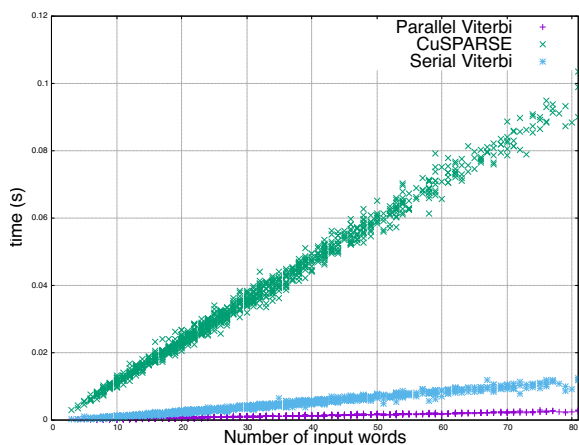


Figure 4: Viterbi decoding times for 1000 individual test sentences compared for our serial, parallel, and cuSPARSE implementations (Titan X).

Our serial implementation Algorithm 1 for Viterbi and Algorithm 2 for forward-backward.

cuSPARSE was used to implement the forward algorithm, using CSR format instead of COO for transition matrices. Since we can't redefine addition and multiplication, we could not implement the Viterbi algorithm. To avoid underflow, we rescaled the vector of forward values at each time step and kept track of the log of the scale in a separate variable.

To be fair, it should be noted that Carmel and OpenFST are much more general than the other implementations listed here. Both perform FST composition in order to decode an input string adding another layer of complexity to the process. The timings for OpenFST and Carmel on Table 2 include composition

7.3 Results

Table 2 shows the overall performance of our Viterbi algorithm and the baseline algorithms. Our parallel implementation does worse than our serial implementation when the transducer used is small (presumably due to the overhead of kernel launches and memory copies), but the speedups increase as the size of the transducer grows, reaching a speedup of 5x. The forward-backward algorithm with expected counts obtains a 5x speedup over the serial code on the largest transducer (See Table 3).

CuSPARSE does significantly worse than even our serial implementation; presumably, it would have done better if the transition matrices of our

transducers were sparser.

Figure 4 shows decoding times for three algorithms (our serial and parallel Viterbi, and cuSPARSE forward) on individual sentences. It can be seen that all three algorithms are roughly linear in the sentence length.

Viterbi is faster than either the forward or backward algorithm across the board. This is because the latter need to add log-probabilities (lines 6 and 13 of Algorithm 2), which involves expensive calls to transcendental functions.

7.4 Comparison across GPU architectures

Table 2 compares the performance of the Kepler-based K40, where we did most of our experiments, with the Maxwell-based Titan X and the Pascal-based Tesla P100. The performance improvement is due to different factors, such as a larger number of active thread blocks per streaming multiprocessor on a GPU architecture, the grid and block size selected to run the kernels, and memory management on the GPU. After the release of the Kepler architecture, the Maxwell architecture introduced an improved workload balancing, reduced arithmetic latency, and faster atomic operations. The Pascal architecture allows speedups over all the other architectures by introducing an increased floating point performance, faster data movement performance (NVLink), larger and more efficient shared memory, and improved atomic operations. Also, SMs on the pascal architecture are more efficient allowing speedups larger speedups than its predecessors. Our parallel implementations were compiled using architecture specific flags (`-arch=compute_XX`) to take full advantage of the architectural enhancements described in this section.

7.5 Comparison against a multi-core implementation

Table 2 shows how our parallel implementation on a GPU compares against a multi-core version of our serial Viterbi algorithm implemented in MPI. We chose MPI since it supports distributed and shared memory unlike OpenMP that supports shared memory only. Results show that a multi-core implementation of the algorithm leads to slower performance than the serial code due to the communication and synchronization overhead. Several cores must transfer information frequently and synchronize all messages on a single core. GPUs perform better than multi-core in this case

Method	Hardware	Training size (lines)							
		1000		10000		100000		150000	
		Time	Ratio	Time	Ratio	Time	Ratio	Time	Ratio
OpenFST	Core i7	42.06	1.9	2547	87	313800	4085	626700	6093
Carmel	Core i7	195.9	9.1	7652	263	224500	2923	376400	3659
our serial	Core i7	10.99	0.5	44.16	1.5	374.2	4.9	534.9	5.2
our serial	Xeon E5	10.84	0.5	42.05	1.4	375.3	4.9	529.6	5.1
our MPI	4-core Core i7	194.0	9.0	581.2	20	1849	24	2243	21
our parallel	K40	27.52	1.3	38.26	1.3	116.5	1.5	131.1	1.3
our parallel	Titan X	25.05	1.2	33.92	1.2	94.07	1.2	121.6	1.2
our parallel	Tesla P100	21.49	1.0	29.04	1.0	76.79	1.0	102.9	1.0

Table 2: Our GPU implementation of the Viterbi algorithm outperforms all others tested on the medium and large FSTs. Times (in seconds) are for decoding a set of 100 examples 1000 times using Viterbi. Ratios are relative to our parallel algorithm on the Tesla P100.

method	Training size (lines)			
	1k	10k	100k	150k
cuSPARSE forward	646	1846	3555	5948
serial forward	36	251	2297	3346
parallel forward	17	37	236	327
serial backward	13	248	3585	5303
parallel backward	43	80	644	1070
serial combined	47	534	6065	8790
parallel combined	60	120	1111	1773

Table 3: Our GPU implementations of the forward and backward algorithms, and forward+backward+expected counts combined, outperform all others tested, on the medium and large FSTs. Times (in seconds) are for processing 100 examples 1000 times, on a Core i7 and K40.

Training size	States	Transitions	Non-zero
1000	3505	443527	3.6%
10000	11644	6792487	5.0%
100000	33125	95381368	8.7%
150000	39420	150971615	9.7%

Table 4: FST Comparison. This table shows the number of states, edges, and percent of non zero elements of the transducers created using 1k/10k/100k/150k examples.

since all the memory is already on the graphics card and the cost of using global memory on the GPU is lower than synchronizing and sharing data between cores.

8 Conclusion

We have shown that our algorithm outperforms several serial implementations (our own serial implementation on a Intel Core i7 and Xeon E machines, Carmel and OpenFST) as well as a GPU implementation using cuSPARSE.

A system with newer and faster cores might achieve higher speedups than a GPU on smaller datasets. However, building a multi-core system that beats a GPU setup can be more expensive. For example, a 16 core Intel Xeon E5-2698 V3 can cost 3,500 USD (Bogoychev and Lopez, 2016). Newer GPU models offer previous generation CPU’s the opportunity to obtain speedups for a lower price (Titan X GPUs sell cheaper than Xeon E5 setups at US\$1,200). Speeding up computation on a GPU would allow users to speed up applications cheaper without investing on a newer multi-core system.

Our implementation has been open-sourced and is available online.³ In the future, we plan to expand this software into a toolkit that includes other algorithms needed to run a full machine translation system.

Acknowledgements

This research was supported in part by a gift of a Tesla K40c GPU card from NVIDIA Corporation.

³<https://bitbucket.org/aargueta2/parallel-decoding>

References

- Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proc. International Conference on Implementation and Application of Automata (CIAA 2007)*, pages 11–23.
- Nathan Bell and Michael Garland. 2008. Efficient sparse matrix-vector multiplication on CUDA. Technical Report NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation.
- Nikolay Bogoychev and Adam Lopez. 2016. N-gram language models for massively parallel devices. In *Proc. ACL*, pages 1944–1953.
- John Canny, David Hall, and Dan Klein. 2013. A multi-teraflop constituency parser using GPUs. In *Proc. EMNLP*, pages 1898–1907.
- Jike Chong, Ekaterina Gonina, Youngmin Yi, and Kurt Keutzer. 2009. A fully data parallel WFST-based large vocabulary continuous speech recognition on a graphics processing unit. In *Proc. INTERSPEECH*, pages 1183–1186.
- Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. ACL*, pages 1–8.
- David Hall, Taylor Berg-Kirkpatrick, and Dan Klein. 2014. Sparser, better, faster GPU parsing. In *Proc. ACL*, pages 208–217.
- Hua He, Jimmy Lin, and Adam Lopez. 2013. Massively parallel suffix array queries and on-demand phrase extraction for statistical machine translation using gpus. In *Proc. NAACL HLT*, pages 325–334.
- W. Daniel Hillis and Guy L. Steele, Jr. 1986. Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183.
- Jungsuk Kim, Jike Chong, and Ian R Lane. 2012. Efficient on-the-fly hypothesis rescoring in a hybrid gpu/cpu-based large vocabulary continuous speech recognition engine. In *Proc. INTERSPEECH*, pages 1035–1038.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL HLT*, pages 48–54.
- Shankar Kumar, Yonggang Deng, and William Byrne. 2005. A weighted finite state transducer translation template model for statistical machine translation. *J. Natural Language Engineering*, 12(1):35–75.
- Richard E. Ladner and Michael J. Fischer. 1980. Parallel prefix computation. *J. ACM*, 27(4):831–838.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289.
- Rongchun Li, Yong Dou, and Dan Zou. 2014. Efficient parallel implementation of three-point viterbi decoding algorithm on CPU, GPU, and FPGA. *Concurrency and Computation: Practice and Experience*, 26(3):821–840.
- Duane Merrill, Michael Garland, and Andrew Grimshaw. 2012. Scalable GPU graph traversal. In *Proc. 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 117–128.
- Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88.
- Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 213–254. Springer.
- Todd Mytkowicz, Madanlal Musuvathi, and Wolfram Schulte. 2014. Data-parallel finite-state machines. In *Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March.
- Veynu Narasiman, Michael Shebanow, Chang Joo Lee, Rustam Miftakhutdinov, Onur Mutlu, and Yale N Patt. 2011. Improving GPU performance via large warps and two-level warp scheduling. In *Proc. IEEE/ACM International Symposium on Microarchitecture*, pages 308–317.
- Hao Peng, Rongke Liu, Yi Hou, and Ling Zhao. 2016. A Gb/s parallel block-based viterbi decoder for convolutional codes on gpu. *arXiv preprint arXiv:1608.00066*.
- Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv*, abs/1605.02688, May.

Learning to Translate in Real-time with Neural Machine Translation

Jiatao Gu[†], Graham Neubig[◇], Kyunghyun Cho[‡] and Victor O.K. Li[†]

[†]The University of Hong Kong [◇]Carnegie Mellon University [‡]New York University

[†]{jiataogu, vli}@eee.hku.hk [◇]gneubig@cs.cmu.edu

[‡]kyunghyun.cho@nyu.edu

Abstract

Translating in real-time, a.k.a. simultaneous translation, outputs translation words before the input sentence ends, which is a challenging problem for conventional machine translation methods. We propose a neural machine translation (NMT) framework for simultaneous translation in which an agent learns to make decisions on when to translate from the interaction with a pre-trained NMT environment. To trade off quality and delay, we extensively explore various targets for delay and design a method for beam-search applicable in the simultaneous MT setting. Experiments against state-of-the-art baselines on two language pairs demonstrate the efficacy of the proposed framework both quantitatively and qualitatively.¹

1 Introduction

Simultaneous translation, the task of translating content in real-time as it is produced, is an important tool for real-time understanding of spoken lectures or conversations (Fügen et al., 2007; Bangalore et al., 2012). Different from the typical machine translation (MT) task, in which translation quality is paramount, simultaneous translation requires balancing the trade-off between translation quality and time delay to ensure that users receive translated content in an expeditious manner (Mieno et al., 2015). A number of methods have been proposed to solve this problem, mostly in the context of phrase-based machine translation. These methods are based on a *segmenter*, which receives the input one word at a time, then decides when to send it to a MT system that translates each

¹Code and data can be found at <https://github.com/nyu-dl/dl4mt-simul-trans>.

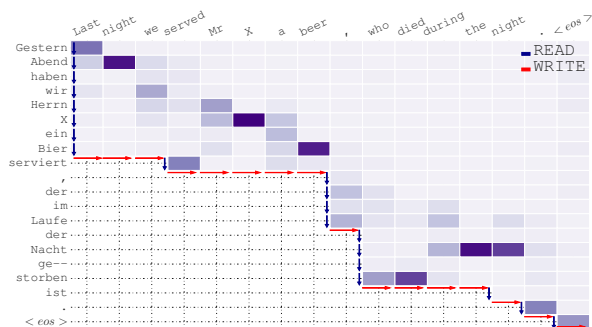


Figure 1: Example output from the proposed framework in DE \rightarrow EN simultaneous translation. The heat-map represents the soft alignment between the incoming source sentence (left, up-to-down) and the emitted translation (top, left-to-right). The length of each column represents the number of source words being waited for before emitting the translation. Best viewed when zoomed digitally.

segment independently (Oda et al., 2014) or with a minimal amount of language model context (Bangalore et al., 2012).

Independently of simultaneous translation, accuracy of standard MT systems has greatly improved with the introduction of neural-network-based MT systems (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014). Very recently, there have been a few efforts to apply NMT to simultaneous translation either through heuristic modifications to the decoding process (Cho and Esipova, 2016), or through the training of an independent segmentation network that chooses when to perform output using a standard NMT model (Satija and Pineau, 2016). However, the former model lacks a capability to learn the appropriate timing with which to perform translation, and the latter model uses a standard NMT model as-is, lacking a holistic design of the modeling and learning within the simultaneous MT context. In addition, neither model has demonstrated gains over previ-

ous segmentation-based baselines, leaving questions of their relative merit unresolved.

In this paper, we propose a unified design for learning to perform neural simultaneous machine translation. The proposed framework is based on formulating translation as an interleaved sequence of two actions: READ and WRITE. Based on this, we devise a model connecting the NMT system and these READ/WRITE decisions. An example of how translation is performed in this framework is shown in Fig. 1, and detailed definitions of the problem and proposed framework are described in §2 and §3. To learn which actions to take when, we propose a reinforcement-learning-based strategy with a reward function that considers both quality and delay (§4). We also develop a beam-search method that performs search within the translation segments (§5).

We evaluate the proposed method on English-Russian (EN-RU) and English-German (EN-DE) translation in both directions (§6). The quantitative results show strong improvements compared to both the NMT-based algorithm and a conventional segmentation methods. We also extensively analyze the effectiveness of the learning algorithm and the influence of the trade-off in the optimization criterion, by varying a target delay. Finally, qualitative visualization is utilized to discuss the potential and limitations of the framework.

2 Problem Definition

Suppose we have a buffer of input words $X = \{x_1, \dots, x_{T_s}\}$ to be translated in real-time. We define the simultaneous translation task as sequentially making two interleaved decisions: READ or WRITE. More precisely, the translator READs a source word x_η from the input buffer in chronological order as translation context, or WRITES a translated word y_τ onto the output buffer, resulting in output sentence $Y = \{y_1, \dots, y_{T_t}\}$, and action sequence $A = \{a_1, \dots, a_T\}$ consists of T_s READs and T_t WRITES, so $T = T_s + T_t$.

Similar to standard MT, we have a measure $Q(Y)$ to evaluate the translation quality, such as BLEU score (Papineni et al., 2002). For simultaneous translation we are also concerned with the fact that each action incurs a time delay $D(A)$. $D(A)$ will mainly be influenced by delay caused by READ, as this entails waiting for a human speaker to continue speaking (about 0.3s per word for an average speaker), while WRITE consists of generating a few words from a machine transla-

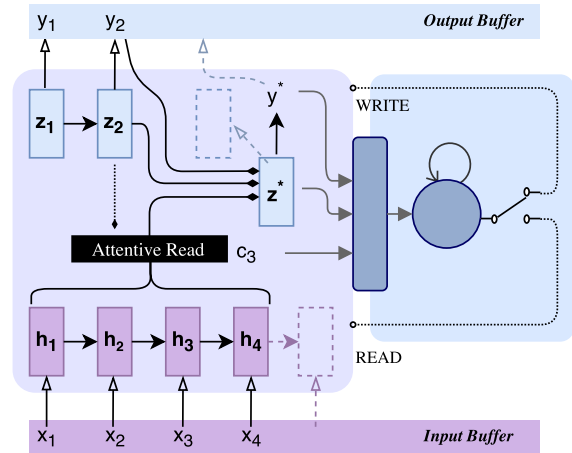


Figure 2: Illustration of the proposed framework: at each step, the NMT environment (left) computes a candidate translation. The recurrent agent (right) will the observation including the candidates and send back decisions—READ or WRITE.

tion system, which is possible on the order of milliseconds. Thus, our objective is finding an optimal policy that generates decision sequences with a good trade-off between higher quality $Q(Y)$ and lower delay $D(A)$. We elaborate on exactly how to define this trade-off in §4.2.

In the following sections, we first describe how to connect the READ/WRITE actions with the NMT system (§3), and how to optimize the system to improve simultaneous MT results (§4).

3 Simultaneous Translation with Neural Machine Translation

The proposed framework is shown in Fig. 2, and can be naturally decomposed into two parts: environment (§3.1) and agent (§3.2).

3.1 Environment

Encoder: READ The first element of the NMT system is the encoder, which converts input words $X = \{x_1, \dots, x_{T_s}\}$ into context vectors $H = \{h_1, \dots, h_{T_s}\}$. Standard NMT uses bi-directional RNNs as encoders (Bahdanau et al., 2014), but this is not suitable for simultaneous processing as using a reverse-order encoder requires knowing the final word of the sentence before beginning processing. Thus, we utilize a simple left-to-right unidirectional RNN as our encoder:

$$h_\eta = \phi_{\text{UNI-ENC}}(h_{\eta-1}, x_\eta) \quad (1)$$

Decoder: WRITE Similar with standard MT, we use an attention-based decoder. In contrast, we

only reference the words that have been read from the input when generating each target word:

$$\begin{aligned} c_\tau^\eta &= \phi_{\text{ATT}}(z_{\tau-1}, y_{\tau-1}, H^\eta) \\ z_\tau^\eta &= \phi_{\text{DEC}}(z_{\tau-1}, y_{\tau-1}, c_\tau^\eta) \\ p(y|y_{<\tau}, H^\eta) &\propto \exp[\phi_{\text{OUT}}(z_\tau^\eta)], \end{aligned} \quad (2)$$

where for τ , $z_{\tau-1}$ and $y_{\tau-1}$ represent the previous decoder state and output word, respectively. H^η is used to represent the incomplete input states, where H^η is a prefix of H . As the WRITE action calculates the probability of the next word on the fly, we need greedy decoding for each step:

$$y_\tau^\eta = \arg \max_y p(y|y_{<\tau}, H^\eta) \quad (3)$$

Note that y_τ^η, z_τ^η corresponds to H^η and is the candidate for y_τ, z_τ . The agent described in the next section decides whether to take this candidate or wait for better predictions.

3.2 Agent

A trainable agent is designed to make decisions $A = \{a_1, \dots, a_T\}$, $a_t \in \mathcal{A}$ sequentially based on observations $O = \{o_1, \dots, o_T\}$, $o_t \in \mathcal{O}$, and then control the translation environment properly.

Observation As shown in Fig 2, we concatenate the current context vector c_τ^η , the current decoder state z_τ^η and the embedding vector of the candidate word y_τ^η as the continuous observation, $o_{\tau+\eta} = [c_\tau^\eta; z_\tau^\eta; E(y_\tau^\eta)]$ to represent the current state.

Action Similarly to prior work (Grissom II et al., 2014), we define the following set of actions:

- **READ:** the agent rejects the candidate and waits to encode the next word from input buffer;
- **WRITE:** the agent accepts the candidate and emits it as the prediction into output buffer;

Policy How the agent chooses the actions based on the observation defines the policy. In our setting, we utilize a stochastic policy π_θ parameterized by a recurrent neural network, that is:

$$\begin{aligned} s_t &= f_\theta(s_{t-1}, o_t) \\ \pi_\theta(a_t|a_{<t}, o_{\leq t}) &\propto g_\theta(s_t), \end{aligned} \quad (4)$$

where s_t is the internal state of the agent, and is updated recurrently yielding the distribution of the action a_t . Based on the policy of our agent, the overall algorithm of greedy decoding is shown in Algorithm 1, The algorithm outputs the translation result and a sequence of observation-action pairs.

Algorithm 1 Simultaneous Greedy Decoding

Require: NMT system ϕ , policy π_θ , τ_{MAX} , input buffer X , output buffer Y , state buffer S .

```

1: Init  $x_1 \leftarrow X, h_1 \leftarrow \phi_{\text{ENC}}(x_1), H^1 \leftarrow \{h_1\}$ 
2:    $z_0 \leftarrow \phi_{\text{INIT}}(H^1), y_0 \leftarrow \langle s \rangle$ 
3:    $\tau \leftarrow 0, \eta \leftarrow 1$ 
4: while  $\tau < \tau_{\text{MAX}}$  do
5:    $t \leftarrow \tau + \eta$ 
6:    $y_\tau^\eta, z_\tau^\eta, o_t \leftarrow \phi(z_{\tau-1}, y_{\tau-1}, H^\eta)$ 
7:    $a_t \sim \pi_\theta(a_t; a_{<t}, o_{<t}), S \leftarrow (o_t, a_t)$ 
8:   if  $a_t = \text{READ}$  and  $x_\eta \neq \langle /s \rangle$  then
9:      $x_{\eta+1} \leftarrow X, h_{\eta+1} \leftarrow \phi_{\text{ENC}}(h_\eta, x_{\eta+1})$ 
10:     $H^{\eta+1} \leftarrow H^\eta \cup \{h_{\eta+1}\}, \eta \leftarrow \eta + 1$ 
11:    if  $|Y| = 0$  then  $z_0 \leftarrow \phi_{\text{INIT}}(H^\eta)$ 
12:  else if  $a_t = \text{WRITE}$  then
13:     $z_\tau \leftarrow z_\tau^\eta, y_\tau \leftarrow y_\tau^\eta$ 
14:     $Y \leftarrow y_\tau, \tau \leftarrow \tau + 1$ 
15:    if  $y_\tau = \langle /s \rangle$  then break

```

4 Learning

The proposed framework can be trained using reinforcement learning. More precisely, we use policy gradient algorithm together with variance reduction and regularization techniques.

4.1 Pre-training

We need an NMT environment for the agent to explore and use to generate translations. Here, we simply pre-train the NMT encoder-decoder on full sentence pairs with maximum likelihood, and assume the pre-trained model is still able to generate reasonable translations even on incomplete source sentences. Although this is likely sub-optimal, our NMT environment based on uni-directional RNNs can treat incomplete source sentences in a manner similar to shorter source sentences and has the potential to translate them more-or-less correctly.

4.2 Reward Function

The policy is learned in order to increase a reward for the translation. At each step the agent will receive a reward signal r_t based on (o_t, a_t) . To evaluate a good simultaneous machine translation, a reward must consider both quality and delay.

Quality We evaluate the translation quality using metrics such as BLEU (Papineni et al., 2002). The BLEU score is defined as the weighted geometric average of the modified n-gram precision BLEU⁰, multiplied by the brevity penalty BP to punish a short translation. In practice, the vanilla

BLEU score is not a good metric at sentence level because being a geometric average, the score will reduce to zero if one of the precisions is zero. To avoid this, we used a smoothed version of BLEU for our implementation (Lin and Och, 2004).

$$\text{BLEU}(Y, Y^*) = \text{BP} \cdot \text{BLEU}^0(Y, Y^*), \quad (5)$$

where Y^* is the reference and Y is the output. We decompose BLEU and use the difference of partial BLEU scores as the reward, that is:

$$r_t^Q = \begin{cases} \Delta \text{BLEU}^0(Y, Y^*, t) & t < T \\ \text{BLEU}(Y, Y^*) & t = T \end{cases} \quad (6)$$

where Y^t is the cumulative output at t ($Y^0 = \emptyset$), and $\Delta \text{BLEU}^0(Y, Y^*, t) = \text{BLEU}^0(Y^t, Y^*) - \text{BLEU}^0(Y^{t-1}, Y^*)$. Obviously, if $a_t = \text{READ}$, no new words are written into Y , yielding $r_t^Q = 0$. Note that we do not multiply BP until the end of the sentence, as it would heavily penalize partial translation results.

Delay As another critical feature, delay judges how much time is wasted waiting for the translation. Ideally we would directly measure the actual time delay incurred by waiting for the next word. For simplicity, however, we suppose it consumes the same amount of time listening for one more word. We define two measurements, global and local, respectively:

- **Average Proportion (AP):** following the definition in (Cho and Esipova, 2016), X, Y are the source and decoded sequences respectively, and we use $s(\tau)$ to denote the number of source words been waited when decoding word y_τ ,

$$0 < d(X, Y) = \frac{1}{|X||Y|} \sum_{\tau} s(\tau) \leq 1 \quad (7)$$

$$d_t = \begin{cases} 0 & t < T \\ d(X, Y) & t = T \end{cases}$$

d is a global delay metric, which defines the average waiting proportion of the source sentence when translating each word.

- **Consecutive Wait length (CW):** in speech translation, listeners are also concerned with long silences during which no translation occurs. To capture this, we also consider on how many words were waited for (READ) consecutively between translating two words. For each action, where we initially define $c_0 = 0$,

$$c_t = \begin{cases} c_{t-1} + 1 & a_t = \text{READ} \\ 0 & a_t = \text{WRITE} \end{cases} \quad (8)$$

- **Target Delay:** We further define ‘‘target delay’’ for both d and c as d^* and c^* , respectively, as different simultaneous translation applications may have different requirements on delay. In our implementation, the reward function for delay is written as:

$$r_t^D = \alpha \cdot [\text{sgn}(c_t - c^*) + 1] + \beta \cdot [d_t - d^*]_+ \quad (9)$$

where $\alpha \leq 0, \beta \leq 0$.

Trade-off between quality and delay A good simultaneous translation system requires balancing the trade-off of translation quality and time delay. Obviously, achieving the best translation quality and the shortest translation delays are in a sense contradictory. In this paper, the trade-off is achieved by balancing the rewards $r_t = r_t^Q + r_t^D$ provided to the system, that is, by adjusting the coefficients α, β and the target delay d^*, c^* in Eq. 9.

4.3 Reinforcement Learning

Policy Gradient We freeze the pre-trained parameters of an NMT model, and train the agent using the policy gradient (Williams, 1992). The policy gradient maximizes the following expected cumulative future rewards, $J = \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^T r_t \right]$, whose gradient is

$$\nabla_{\theta} J = \mathbb{E}_{\pi_\theta} \left[\sum_{t'=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{t'} | \cdot) R_t \right] \quad (10)$$

$R_t = \sum_{k=t}^T [r_k^Q + r_k^D]$ is the cumulative future rewards for current observation and action. In practice, Eq. 10 is estimated by sampling multiple action trajectories from the current policy π_θ , collecting the corresponding rewards.

Variance Reduction Directly using the policy gradient suffers from high variance, which makes learning unstable and inefficient. We thus employ the variance reduction techniques suggested by Mnih and Gregor (2014). We subtract from R_t the output of a baseline network b_φ to obtain $\hat{R}_t = R_t - b_\varphi(o_t)$, and centered re-scale the reward as $\tilde{R}_t = \frac{\hat{R}_t - b}{\sqrt{\sigma^2 + \epsilon}}$ with a running average b and standard deviation σ . The baseline network is trained to minimize the squared loss as follows:

$$L_\varphi = \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^T \|R_t - b_\varphi(o_t)\|^2 \right] \quad (11)$$

We also regularize the negative entropy of the policy to facilitate exploration.

Algorithm 2 Learning with Policy Gradient

Require: NMT system ϕ , agent θ , baseline φ

- 1: Pretrain the NMT system ϕ using MLE;
 - 2: Initialize the agent θ ;
 - 3: **while** stopping criterion fails **do**
 - 4: Obtain a translation pairs: $\{(X, Y^*)\}$;
 - 5: **for** $(Y, S) \sim \text{Simultaneous Decoding}$ **do**
 - 6: **for** (o_t, a_t) in S **do**
 - 7: Compute the quality: r_t^Q ;
 - 8: Compute the delay: r_t^D ;
 - 9: Compute the baseline: $b_\varphi(o_t)$;
 - 10: Collect the future rewards: $\{R_t\}$;
 - 11: Perform variance reduction: $\{\tilde{R}_t\}$;
 - 12: Update: $\theta \leftarrow \theta + \lambda_1 \nabla_\theta [J - \kappa \mathcal{H}(\pi_\theta)]$
 - 13: Update: $\varphi \leftarrow \varphi - \lambda_2 \nabla_\varphi L$
-

The overall learning algorithm is summarized in Algorithm 2. For efficiency, instead of updating with stochastic gradient descent (SGD) on a single sentence, both the agent and the baseline are optimized using a minibatch of multiple sentences.

5 Simultaneous Beam Search

In previous sections we described a simultaneous greedy decoding algorithm. In standard NMT it has been shown that beam search, where the decoder keeps a beam of k translation trajectories, greatly improves translation quality (Sutskever et al., 2014), as shown in Fig. 3 (A).

It is non-trivial to directly apply beam-search in simultaneous machine translation, as beam search waits until the last word to write down translation. Based on our assumption WRITE does not cost delay, we can perform a simultaneous beam-search when the agent chooses to consecutively WRITE: keep multiple beams of translation trajectories in temporary buffer and output the best path when the agent switches to READ. As shown in Fig. 3 (B) & (C), it tries to search for a relatively better path while keeping the delay unchanged.

Note that we do not re-train the agent for simultaneous beam-search. At each step we simply input the observation of the current best trajectory into the agent for making next decision.

6 Experiments

6.1 Settings

Dataset To extensively study the proposed simultaneous translation model, we train and evaluate it on two different language pairs: “English-

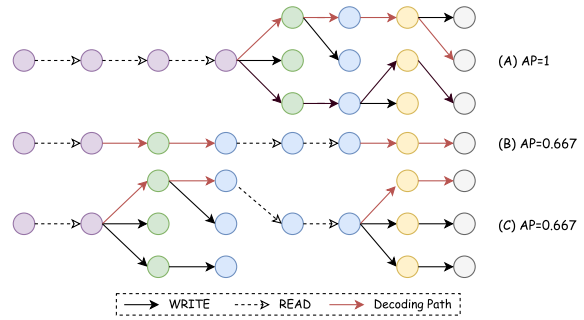


Figure 3: Illustrations of (A) beam-search, (B) simultaneous greedy decoding and (C) simultaneous beam-search.

German (EN-DE)” and “English-Russian (EN-RU)” in both directions per pair. We use the parallel corpora available from WMT’15² for both pre-training the NMT environment and learning the policy. We utilize newstest-2013 as the validation set to evaluate the proposed algorithm. Both the training set and the validation set are tokenized and segmented into sub-word units with byte-pair encoding (BPE) (Sennrich et al., 2015). We only use sentence pairs where both sides are less than 50 BPE subword symbols long for training.

Environment & Agent Settings We pre-trained the NMT environments for both language pairs and both directions following the same setting from (Cho and Esipova, 2016). We further built our agents, using a recurrent policy with 512 GRUs and a softmax function to produce the action distribution. All our agents are trained using policy gradient using Adam (Kingma and Ba, 2014) optimizer, with a mini-batch size of 10. For each sentence pair in a batch, 5 trajectories are sampled. For testing, instead of sampling we pick the action with higher probability each step.

Baselines We compare the proposed methods against previously proposed baselines. For fair comparison, we use the same NMT environment:

- **Wait-Until-End (WUE):** an agent that starts to WRITE only when the last source word is seen. In general, we expect this to achieve the best quality of translation. We perform both greedy decoding and beam-search with this method.
- **Wait-One-Step (WOS):** an agent that WRITES after each READS. Such a policy is problematic when the source and target language pairs have different word orders or lengths (e.g. EN-DE).

²<http://www.statmt.org/wmt15/>

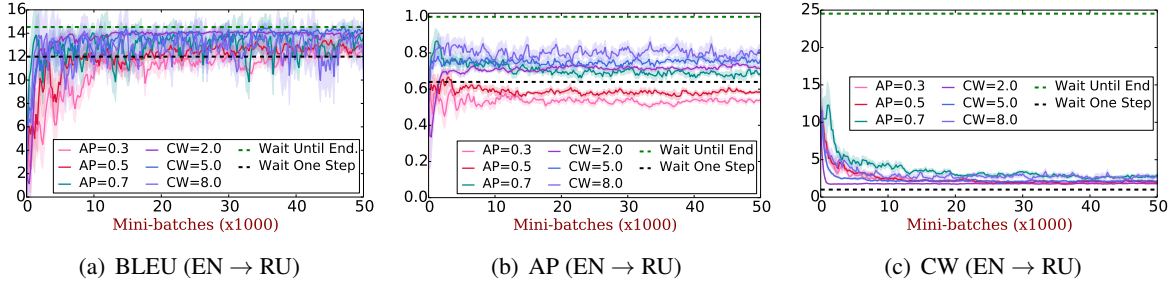


Figure 4: Learning progress curves for variant delay targets on the validation dataset for EN \rightarrow RU. Every time we only keep one target for one delay measure. For instance when using target AP, the coefficient of α in Eq. 9 will be set 0.

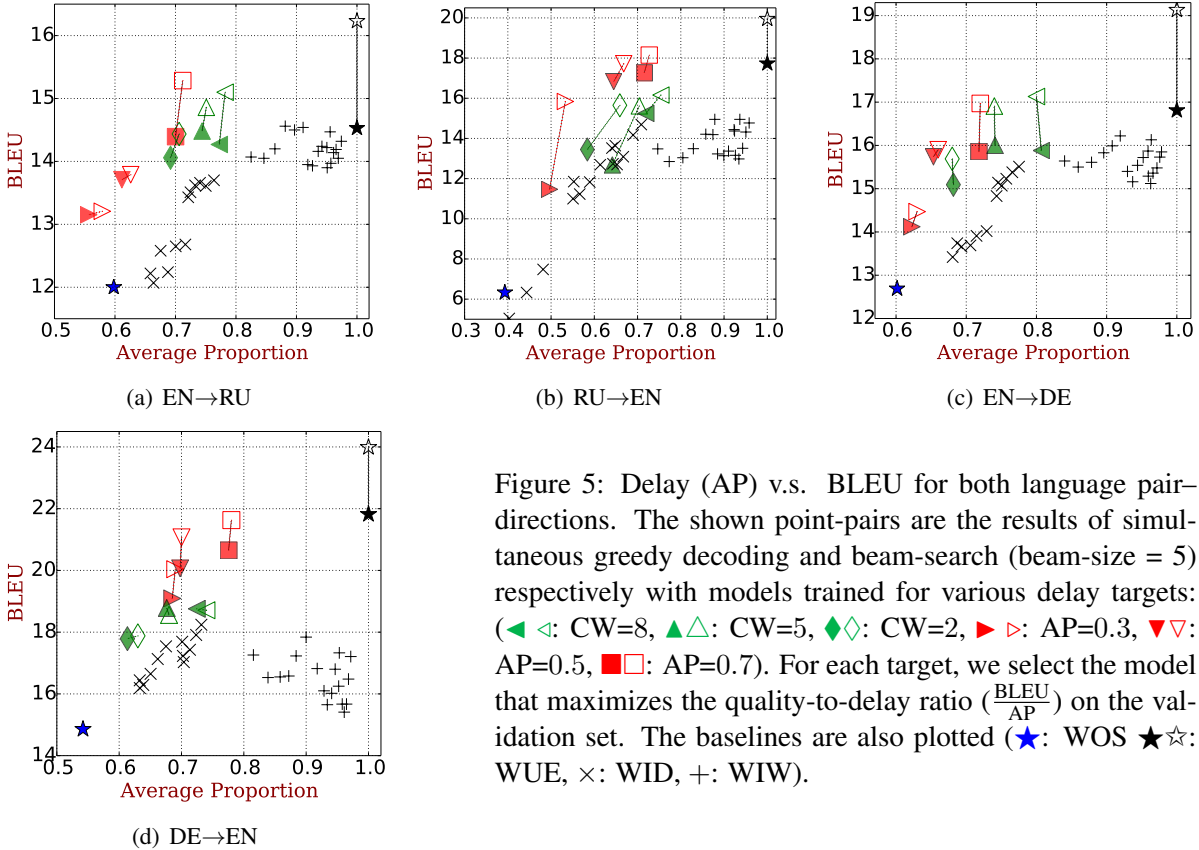


Figure 5: Delay (AP) v.s. BLEU for both language pair-directions. The shown point-pairs are the results of simultaneous greedy decoding and beam-search (beam-size = 5) respectively with models trained for various delay targets: (\blacktriangleleft , \triangleleft : CW=8, \blacktriangle , \triangle : CW=5, \blacklozenge , \lozenge : CW=2, \blacktriangleright , \triangleright : AP=0.3, \blacktriangledown , \triangledown : AP=0.5, \blacksquare , \square : AP=0.7). For each target, we select the model that maximizes the quality-to-delay ratio ($\frac{\text{BLEU}}{\text{AP}}$) on the validation set. The baselines are also plotted (\star : WOS \star : WUE, \times : WID, $+$: WIW).

- **Wait-If-Worse/Wait-If-Diff (WIW/WID)**: as proposed by Cho and Esipova (2016), the algorithm first pre-READS the next source word, and accepts this READ when the probability of the most likely target word decreases (WIW), or the most likely target word changes (WID).
- **Segmentation-based (SEG)** (Oda et al., 2014): a state-of-the-art segmentation-based algorithm based on optimizing segmentation to achieve the highest quality score. In this paper, we tried the simple greedy method (SEG1) and the greedy method with POS Constraint (SEG2).

6.2 Quantitative Analysis

In order to evaluate the effectiveness of our reinforcement learning algorithms with different re-

ward functions, we vary the target delay $d^* \in \{0.3, 0.5, 0.7\}$ and $c^* \in \{2, 5, 8\}$ for Eq. 9 separately, and trained agents with α and β adjusted to values that provided stable learning for each language pair according to the validation set.

Learning Curves As shown in Fig. 4, we plot learning progress for EN-RU translation with different target settings. It clearly shows that our algorithm effectively increases translation quality for all the models, while pushing the delay close, if not all of the way, to the target value. It can also be noted from Fig. 4 (a) and (b) that there exists strong correlation between the two delay measures, implying the agent can learn to decrease both AP and CW simultaneously.

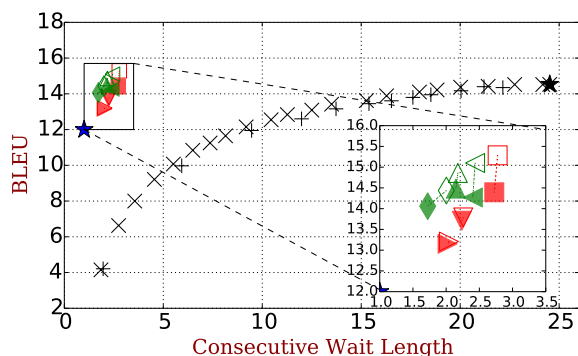


Figure 6: Delay (CW) v.s. BLEU score for EN \rightarrow RU, (\triangleleft : CW=8, \triangle : CW=5, \diamond : CW=2, \blacktriangleright : AP=0.3, \blacktriangledown : AP=0.5, \blacksquare : AP=0.7), against the baselines (\star : WOS \star : WUE, +: SEG1, \times : SEG2).

Quality v.s. Delay As shown in Fig. 5, it is clear that the trade-off between translation quality and delay has very similar behaviors across both language pairs and directions. The smaller delay (AP or CW) the learning algorithm is targeting, the lower quality (BLEU score) the output translation. It is also interesting to observe that, it is more difficult for “ \rightarrow EN” translation to achieve a lower AP target while maintaining good quality, compared to “EN \rightarrow ”. In addition, the models that are optimized on AP tend to perform better than those optimized on CW, especially in “ \rightarrow EN” translation. German and Russian sentences tend to be longer than English, hence require more consecutive waits before being able to emit the next English symbol.

v.s. Baselines In Fig. 5 and 6, the points closer to the upper left corner achieve better trade-off performance. Compared to WUE and WOS which can ideally achieve the best quality (but the worst delay) and the best delay (but poor quality) respectively, all of our proposed models find a good balance between quality and delay. Some of the proposed models can achieve good BLEU scores close to WUE, while have much smaller delay.

Compared to the method of Cho and Esipova (2016) based on two hand-crafted rules (WID, WIW), in most cases our proposed models find better trade-off points, while there are a few exceptions. We also observe that the baseline models have trouble controlling the delay in a reasonable area. In contrast, by optimizing towards a given target delay, our proposed model is stable while maintaining good translation quality.

We also compared against Oda et al. (2014)’s

state-of-the-art segmentation algorithm (SEG). As shown in Fig 6, it is clear that although SEG can work with variant segmentation lengths (CW), the proposed model outputs high quality translations at a much smaller CW. We conjecture that this is due to the independence assumption in SEG, while the RNNs and attention mechanism in our model makes it possible to look at the whole history to decide each translated word.

w/o Beam-Search We also plot the results of simultaneous beam-search instead of using greedy decoding. It is clear from Fig. 5 and 6 that most of the proposed models can achieve an visible increase in quality together with a slight increase in delay. This is because beam-search can help to avoid bad local minima. We also observe that the simultaneous beam-search cannot bring as much improvement as it did in the standard NMT setting. In most cases, the smaller delay the model achieves, the less beam search can help as it requires longer consecutive WRITE segments for extensive search to be necessary. One possible solution is to consider the beam uncertainty in the agent’s READ/WRITE decisions. We leave this to future work.

6.3 Qualitative Analysis

In this section, we perform a more in-depth analysis using examples from both EN-RU and EN-DE pairs, in order to have a deeper understanding of the proposed algorithm and its remaining limitations. We only perform greedy decoding to simplify visualization.

EN \rightarrow RU As shown in Fig 8, since both English and Russian are Subject-Verb-Object (SVO) languages, the corresponding words may share the same order in both languages, which makes simultaneous translation easier. It is clear that the larger the target delay (AP or CW) is set, the more words are read before translating the corresponding words, which in turn results in better translation quality. We also note that very early WRITE commonly causes bad translation. For example, for AP=0.3 & CW=2, both the models choose to WRITE in the very beginning the word “The”, which is unreasonable since Russian has no articles, and there is no word corresponding to it. One good feature of using NMT is that the more words the decoder READS, the longer history is saved, rendering simultaneous translation easier.

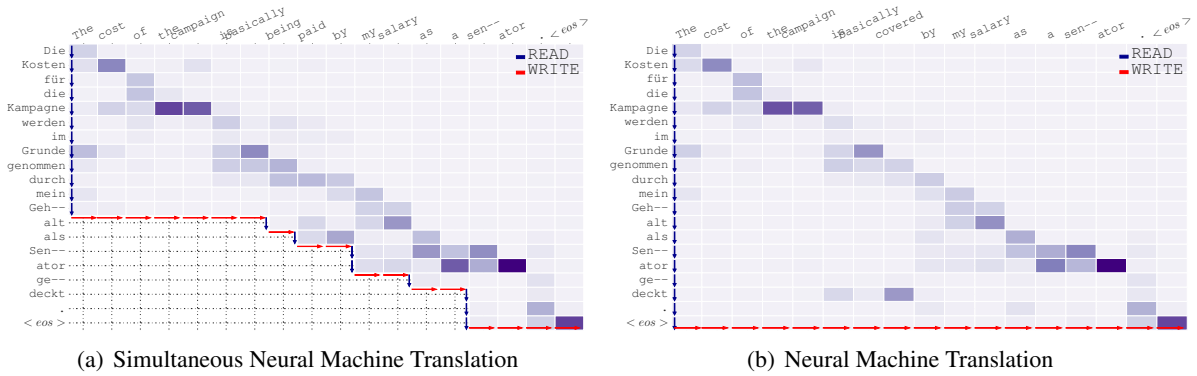


Figure 7: Comparison of DE→EN examples using the proposed framework and usual NMT system respectively. Both the heatmaps share the same setting with Fig. 1. The verb “gedeckt” is incorrectly translated in simultaneous translation.

Source	AP=0.3	AP=0.7	CW=2	CW=8
The	The The		The	
people	p-- i-- ent the p-- ol-- s		p-- riv--	
,	,		,	
as	as		ers	
I	я слышал	Люди		Люди
heard	я слышал	как я слышал		
in			как	
the			я слышал	
countryside	в	в		
,				
want	сельской местности	сельской	в	как я
a		местности		слышал
Government			сельской	
that				в
is			местности	сельской
not				
made	хочу			
up	правительство			местности
of			хочу	
thi--				
eves		хотят	правительство	
.				
<eos>	которое не производится во-- ров .	, чтобы правительство , которое не в-- меши-- вается в во-- ры .	, которое не является состав-- ной частью во-- ров .	хотят , чтобы правительство , которое не в-- меши-- вается в во-- ры .
Summary	BLEU=39/ AP=0.46	BLEU=64/AP=0.77	BLEU=54/CW=1.76	BLEU=64/CW=2.55

Figure 8: Given the example input sentence (leftmost column), we show outputs by models trained for various delay targets. For these outputs, each row corresponds to one source word and represents the emitted words (maybe empty) after reading this word. The corresponding source and target words are in the same color for all model outputs.

DE→EN As shown in Fig 1 and 7 (a), where we visualize the attention weights as soft alignment between the progressive input and output sentences, the highest weights are basically along the diagonal line. This indicates that our simultaneous translator works by waiting for enough source words with high alignment weights and then switching to write them.

DE-EN translation is likely more difficult as German usually uses Subject-Object-Verb (SOV) constructions a lot. As shown in Fig 1, when a sentence (or a clause) starts the agent has learned such policy to READ multiple steps to approach the verb (e.g. serviert and gestorben in Fig 1). Such a policy is still limited when the verb is very far from the subject. For instance in Fig. 7, the simultane-

ous translator achieves almost the same translation with standard NMT except for the verb “gedeckt” which corresponds to “covered” in NMT output. Since there are too many words between the verb “gedeckt” and the subject “Kosten für die Kampagne werden”, the agent gives up reading (otherwise it will cause a large delay and a penalty) and WRITES “being paid” based on the decoder’s hypothesis. This is one of the limitations of the proposed framework, as the NMT environment is trained on complete source sentences and it may be difficult to predict the verb that has not been seen in the source sentence. One possible way is to fine-tune the NMT model on incomplete sentences to boost its prediction ability. We will leave this as future work.

7 Related Work

Researchers commonly consider the problem of simultaneous machine translation in the scenario of real-time speech interpretation (Fügen et al., 2007; Bangalore et al., 2012; Fujita et al., 2013; Rangarajan Sridhar et al., 2013; Yarmohammadi et al., 2013). In this approach, the incoming speech stream required to be translated are first recognized and segmented based on an automatic speech recognition (ASR) system. The translation model then works independently based on each of these segments, potentially limiting the quality of translation. To avoid using a fixed segmentation algorithm, Oda et al. (2014) introduced a trainable segmentation component into their system, so that the segmentation leads to better translation quality. Grissom II et al. (2014) proposed a similar framework, however, based on reinforcement learning. All these methods still rely on translating each segment independently without previous context.

Recently, two research groups have tried to apply the NMT framework to the simultaneous translation task. Cho and Esipova (2016) proposed a similar waiting process. However, their waiting criterion is manually defined without learning. Satija and Pineau (2016) proposed a method similar to ours in overall concept, but it significantly differs from our proposed method in many details. The biggest difference is that they proposed to use an agent that passively reads a new word at each step. Because of this, it cannot consecutively decode multiple steps, rendering beam search difficult. In addition, they lack the comparison to any existing approaches. On the other hand, we per-

form an extensive experimental evaluation against state-of-the-art baselines, demonstrating the relative utility both quantitatively and qualitatively.

The proposed framework is also related to some recent efforts about online sequence-to-sequence (SEQ2SEQ) learning. Jaitly et al. (2015) proposed a SEQ2SEQ ASR model that takes fixed-sized segments of the input sequence and outputs tokens based on each segment in real-time. It is trained with alignment information using supervised learning. A similar idea for online ASR is proposed by Luo et al. (2016). Similar to Satija and Pineau (2016), they also used reinforcement learning to decide whether to emit a token while reading a new input at each step. Although sharing some similarities, ASR is very different from simultaneous MT with a more intuitive definition for segmentation. In addition, Yu et al. (2016) recently proposed an online alignment model to help sentence compression and morphological inflection. They regarded the alignment between the input and output sequences as a hidden variable, and performed transitions over the input and output sequence. By contrast, the proposed READ and WRITE actions do not necessarily to be performed on aligned words (e.g. in Fig. 1), and are learned to balance the trade-off of quality and delay.

8 Conclusion

We propose a unified framework to do neural simultaneous machine translation. To trade off quality and delay, we extensively explore various targets for delay and design a method for beam-search applicable in the simultaneous MT setting. Experiments against state-of-the-art baselines on two language pairs demonstrate the efficacy both quantitatively and qualitatively.

Acknowledgments

KC acknowledges the support by Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015-2016). GN acknowledges the support of the Microsoft CORE program. This work was also partly supported by Samsung Electronics (Project: “Development and Application of Larger-Context Neural Machine Translation”).

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 437–445. Association for Computational Linguistics.
- Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.
- Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.
- Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *INTERSPEECH*.
- Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Dont until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352, Doha, Qatar, October. Association for Computational Linguistics.
- Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2015. An online sequence-to-sequence model using partial conditioning. *arXiv preprint arXiv:1511.04868*.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics.
- Yuping Luo, Chung-Cheng Chiu, Navdeep Jaitly, and Ilya Sutskever. 2016. Learning online alignments with continuous rewards policy gradient. *arXiv preprint arXiv:1608.01281*.
- Takashi Mieno, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Speed or accuracy? a study in evaluation of simultaneous speech translation. In *INTERSPEECH*.
- Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.
- Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 551–556, Baltimore, Maryland, June. Association for Computational Linguistics.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.
- Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Atlanta, Georgia, June. Association for Computational Linguistics.
- Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML2016*.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.
- Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *IJCNLP*, pages 1032–1036.
- Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. *arXiv preprint arXiv:1609.08194*.

A Multifaceted Evaluation of Neural versus Phrase-Based Machine Translation for 9 Language Directions

Antonio Toral*
University of Groningen
The Netherlands
a.toral.ruiz@rug.nl

Víctor M. Sánchez-Cartagena
Prompsit Language Engineering
Av. Universitat s/n. Edifici Quorum III
E-03202 Elx, Spain
vmsanchez@prompsit.com

Abstract

We aim to shed light on the strengths and weaknesses of the newly introduced neural machine translation paradigm. To that end, we conduct a multifaceted evaluation in which we compare outputs produced by state-of-the-art neural machine translation and phrase-based machine translation systems for 9 language directions across a number of dimensions. Specifically, we measure the similarity of the outputs, their fluency and amount of reordering, the effect of sentence length and performance across different error categories. We find out that translations produced by neural machine translation systems are considerably different, more fluent and more accurate in terms of word order compared to those produced by phrase-based systems. Neural machine translation systems are also more accurate at producing inflected forms, but they perform poorly when translating very long sentences.

1 Introduction

A new paradigm to statistical machine translation, neural MT (NMT), has emerged very recently and has already surpassed the performance of the mainstream approach in the field, phrase-based MT (PBMT), for a number of language pairs, e.g. (Sennrich et al., 2016b; Luong et al., 2015; Costa-Jussà and Fonollosa, 2016; Chung et al., 2016).

In PBMT (Koehn, 2010) different models (translation, reordering, target language, etc.) are trained independently and combined in a log-linear scheme in which each model is assigned a

different weight by a tuning algorithm. On the contrary, in NMT all the components are jointly trained to maximise translation quality. NMT systems have a strong generalisation power because they encode translation units as numeric vectors that represent concepts, whereas in PBMT translation units are encoded as strings. Moreover, NMT systems are able to model long-distance phenomena thanks to the use of recurrent neural networks, e.g. long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent units (Chung et al., 2014).

The translations produced by NMT systems have been evaluated thus far mostly in terms of overall performance scores, be it by means of automatic or human evaluations. This has been the case of last year's news translation shared task at the First Conference on Machine Translation (WMT16).¹ In this translation task, outputs produced by participant MT systems, the vast majority of which fall under either the phrase-based or neural approaches, were evaluated (i) automatically with the BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) metrics, and (ii) manually by means of ranking translations (Federmann, 2012) and monolingual semantic similarity (Graham et al., 2016). In all these evaluations, the performance of each system is measured by means of an overall score, which, while giving an indication of the general performance of a given system, does not provide any additional information.

In order to understand better the new NMT paradigm and in what respects it provides better (or worse) translation quality than state-of-the-art PBMT, Bentivogli et al. (2016) conducted a detailed analysis for the English-to-German language direction. In a nutshell, they found out that NMT (i) decreases post-editing effort, (ii) de-

*Work partly done at his previous position in Dublin City University, Ireland.

¹<http://www.statmt.org/wmt16/translation-task.html>

grades faster than PBMT with sentence length and (iii) results in a notable improvement regarding re-ordering.

In this paper we delve further in this direction by conducting a multilingual and multifaceted evaluation in order to find answers to the following research questions. Whether, in comparison to PBMT, NMT systems result in:

- considerably different output and higher degree of variability;
- more or less fluent output;
- more or less monotone translations;
- translations with better or worse word order;
- better or worse translations depending on sentence length;
- less or more errors for different error categories: inflectional, reordering and lexical;

Hereunder we specify the main differences and similarities between this work and that of Bentivogli et al. (2016):

- Language directions. They considered 1 while our study comprises 9.
- Content. They dealt with transcribed speeches while we work with news stories. Previous research has shown that these two types of content pose different challenges for MT (Ruiz and Federico, 2014).
- Size of evaluation data. Their test set had 600 sentences while our test sets span from 1 999 to 3 000 depending on the language direction.
- Reference type. Their references were both independent from the MT output and also post-edited, while we have access only to single independent references.
- Analyses. While some analyses overlap, some are novel in our experiments. Namely, output similarity, fluency and degree of re-ordering performed.

Our analyses are conducted on the best PBMT and NMT systems submitted to the WMT16 translation task for each language direction. This (i) guarantees the reproducibility of our results as all the MT outputs are publicly available, (ii) ensures

that the systems evaluated are state-of-the-art, as they are the result of the latest developments at top MT research groups worldwide, and (iii) allows the conclusions that will be drawn to be rather general, as 6 languages from 4 different families (Germanic, Slavic, Romance and Finno-Ugric) are covered in the experiments.

The rest of the paper is organised as follows. Section 2 describes the experimental setup. Subsequent sections cover the experiments carried out in which we measured different aspects of NMT, namely: output similarity (Section 3), fluency (Section 4), degree of reordering and quality of word order (Section 5), sentence length (Section 6), and amount of errors for different error categories (Section 7). Finally, Section 8 holds the conclusions and proposals for future work.

2 Experimental Setup

The experiments are run on the best PBMT and NMT constrained systems submitted to the news translation task of WMT16. We selected such systems according to the human evaluation (Bojar et al., 2016, Sec. 3.4).² We noted that many of the PBMT systems contain neural features, mainly in the form of language models. If the best PBMT submission contains any neural features we use this as the PBMT system in our analyses as long as none of these features is a fully-fledged NMT system. This was the case of the best submission in terms of BLEU for RU→EN (Junczys-Dowmunt et al., 2016).

Out of the 12 language directions at the translation task, we conduct experiments on 9.³ These are the language pairs between English (EN) and Czech (CS), German (DE), Finnish (FI), Romanian (RO) and Russian (RU) in both directions (except for Finnish, where only the EN→FI direction is covered as no NMT system was submitted for the opposite direction, FI→EN). Finally, there was an additional language at the shared task, Turkish, that is not considered here, as either none of the systems submitted was neural (Turkish→EN), or there was one such system but its performance

²When there are not statistically significant differences between two or more NMT or PBMT systems (i.e. they belong to the same equivalence class), we pick the one with the highest BLEU score. If two NMT or PBMT systems were the best according to BLEU (draw), we pick the one with the best TER score.

³Some experiments are run on a subset of these languages due to the lack of required tools for some of the languages involved.

Language Pair	MT Paradigm	System details
EN→CS	PBMT NMT	Phrase-based, word clusters (Ding et al., 2016) Unsupervised word segmentation and backtranslated monolingual corpora (Sennrich et al., 2016a)
EN→DE	hierarchical PBMT NMT	String-to-tree, neural and dependency language models (Williams et al., 2016) Same as for EN→CS
EN→FI	PBMT NMT	Phrase-based, rule-based and unsupervised word segmentation, operation sequence model (Durrani et al., 2011), bilingual neural language model (Devlin et al., 2014), re-ranked with a recurrent neural language model (Sánchez-Cartagena and Toral, 2016) Rule-based word segmentation, backtranslated monolingual corpora (Sánchez-Cartagena and Toral, 2016)
EN→RO	PBMT NMT	Phrased-based, operation sequence model, monolingual and bilingual neural language models (Williams et al., 2016) Same as for EN→CS
EN→RU	PBMT NMT	Phrase-based, word clusters, bilingual neural language model (Ding et al., 2016) Same as for EN→CS
CS→EN	PBMT NMT	Same as for EN→CS Same as for EN→CS
DE→EN	PBMT NMT	Phrase-based, pre-reordering, compound splitting (Williams et al., 2016) Same as for EN→CS plus reranked with a right-to-left model
RO→EN	PBMT NMT	Phrase-based, operation sequence model, monolingual neural language model (Williams et al., 2016) Same as for EN→CS
RU→EN	PBMT NMT	Phrase-based, lemmas in word alignment, sparse features, bilingual neural language model and transliteration (Lo et al., 2016) Same as for EN→CS

Table 1: Details of the best systems pertaining to the PBMT and NMT paradigms submitted to the WMT16 news translation task for each language direction.

was extremely low (EN→Turkish) and hence most probably not representative of the state-of-the-art in NMT.

Table 1 shows the main characteristics of the best PBMT and NMT systems submitted to the WMT16 news translation task. It should be noted that all the NMT systems listed in the table fall under the encoder-decoder architecture with attention (Bahdanau et al., 2015) and operate on subword units. Word segmentation is carried out with the help of a lexicon in the EN→FI direction (Sánchez-Cartagena and Toral, 2016) and in an unsupervised way in the remaining directions (Sennrich et al., 2016a).

2.1 Overall Evaluation

First, and in order to contextualise our analyses below, we report the BLEU scores achieved by the best NMT and PBMT systems for each language direction at WMT16’s news translation task in Table 2.⁴ The best NMT system clearly outperforms the best PBMT system for all language directions out of English (relative improvements range from 5.5% for EN→RO to 17.6% for EN→FI) and the human evaluation (Bojar et al., 2016, Sec. 3.4) confirms these results. In the opposite direction, the human evaluation shows that the best NMT system outperforms the best PBMT system for all language directions except when the source language is Russian. This slightly differs from the automatic evaluation, according to which NMT outperforms PBMT for translations from Czech (3.3% relative improvement) and German (9.9%) but underperforms PBMT for translations from Romanian (-3.7%) and Russian (-3.8%).

3 Output Similarity

The aim of this analysis is to assess to which extent translations produced by NMT systems are different from those produced by PBMT systems. We measure this by taking the outputs of the top n ⁵ NMT and PBMT systems submitted to each language direction and checking their pairwise overlap in terms of the chrF1 (Popović, 2015)

⁴We report the official results from <http://matrix.statmt.org/matrix> for the test set *newstest2016* using normalised BLEU (column z *BLEU-cased-norm*).

⁵The number of systems considered is different for each language direction as it depends on the number of systems submitted. Namely, we have considered 2 NMT and 2 PBMT into Czech, 3 NMT and 5 PBMT into German, 2 NMT and 4 PBMT into Finnish, 2 NMT and 4 PBMT into Romanian and 2 NMT and 3 PBMT into Russian.

System	CS	DE	FI	RO	RU
	From EN				
PBMT	23.7	30.6	15.3	27.4	24.3
NMT	25.9	34.2	18.0	28.9	26.0
	Into EN				
PBMT	30.4	35.2	23.7	35.4	29.3
NMT	31.4	38.7	-	34.1	28.2

Table 2: BLEU scores of the best NMT and PBMT systems for each language pair at WMT16’s news translation task. If the difference between them is statistically significant according to paired bootstrap resampling (Koehn, 2004) with $p = 0.05$ and 1 000 iterations, the highest score is shown in bold.

automatic evaluation metric.⁶ In order to make sure that all systems considered are truly different (rather than different runs of the same system) we consider only 1 system per paradigm (NMT and PBMT) submitted by each team for each language direction.

We would consider NMT outputs considerably different (with respect to PBMT) if they resemble each other (i.e. high pairwise overlap between NMT outputs) more than they do to PBMT systems (i.e. low overlap between an output by NMT and another by PBMT). This analysis is carried out only for language directions out of English, as for all the language directions into English there was, at most, 1 NMT submission.

TL	2 NMT	2 PBMT	NMT & PBMT
CS	68.66	77.63	64.34
DE	72.10	72.97	66.80
FI	56.03	57.42	55.55
RO	69.47	75.96	68.77
RU	35.52	43.35	29.87

Table 3: Average of the overlaps between pairs of outputs produced by the top n NMT and PBMT systems for each language direction from English to the target language (TL). The higher the value, the larger is the overlap.

Table 3 shows the results. We can observe the same trends for all the language directions, namely: (i) the highest overlaps are between pairs

⁶Throughout our analyses we use this metric as it has been shown to correlate better with human judgements than the *de facto* standard automatic metric, BLEU, when the target language is a morphologically rich language such as Finnish, while its correlation is on par with BLEU for languages with simpler morphology such as English (Popović, 2015).

of PBMT systems; (ii) next, we have overlaps between NMT systems; (iii) finally, overlaps between PBMT and NMT are the lowest.

We can conclude then that NMT systems lead to considerably different outputs compared to PBMT. The fact that there is higher inter-system variability in NMT than in PBMT (i.e. overlaps between pairs of NMT systems are lower than between pairs of PBMT systems) may surprise the reader, considering the fact that all NMT systems belong to the same paradigm (encoder-decoder with attention) while for some language directions (EN→DE, EN→FI and EN→RO) there are PBMT systems belonging to two different paradigms (pure phrase-based and hierarchical). However, the higher variability among NMT translations can be attributed, we believe, to the fact that NMT systems use numeric vectors that represent concepts instead of strings as translation units.

4 Fluency

In this experiment we aim to find out whether the outputs produced by NMT systems are more or less fluent than those produced by PBMT systems. To that end, we take perplexity of the MT outputs on neural language models (LMs) as a proxy for fluency. The LMs are built using TheanoLM (Enarvi and Kurimo, 2016). They contain 100 units in the projection layer, 300 units in the LSTM layer, and 300 units in the *tanh* layer, following the setup described by Enarvi and Kurimo (2016, Sec. 3.2). The training algorithm is Adagrad (Duchi et al., 2011) and we used 1 000 word classes obtained with `mkcls` from the training corpus. Vocabulary is limited to the most frequent 50 000 tokens.

LMs are trained on a random sample of 4 million sentences selected from the News Crawl 2015 monolingual corpora, available for all the languages considered.⁷

Table 4 shows the results. For all the language directions considered but one, perplexity is higher on the PBMT output compared to the NMT output. The only exception is translation into Finnish, in which perplexity on the PBMT output is slightly lower, probably because its fluency was improved by reranking it with a neural LM similar to the one

⁷<http://data.statmt.org/wmt16/translation-task/training-monolingual-news-crawl.tgz>

Language direction	PBMT	NMT	Rel. diff.
EN→CS	202.91	173.33	-14.58%
EN→DE	131.54	107.08	-18.60%
EN→FI	214.10	222.40	3.88%
EN→RO	124.66	116.33	-6.68%
EN→RU	158.18	127.83	-19.19%
CS→EN	110.08	102.36	-7.01%
DE→EN	122.26	104.72	-14.35%
RO→EN	106.08	102.18	-3.68%
RU→EN	123.86	106.75	-13.81%
Average	143.74	129.22	-10.45%

Table 4: Perplexity scores for the outputs of the best NMT and PBMT systems on language models built on 4 million sentences randomly selected from the News Crawl 2015 corpora.

we use in this experiment (Sánchez-Cartagena and Toral, 2016). The average relative difference, i.e. considering all language directions, is notable at -10.45%. Thus, our experiment shows that the outputs produced by NMT systems are, in general, more fluent than those produced by PBMT systems.

One may argue that the perplexity obtained for NMT outputs is lower than that for PBMT outputs because the LMs we used to measure perplexity follow the same model as the decoder of the NMT architecture (Bahdanau et al., 2015) and hence perplexity on a neural LM is not a valid proxy for fluency. However, the following facts support our strategy:

- The manual evaluation of fluency carried out at the WMT16 shared translation task (Bogiar et al., 2016, Sec. 3.5) already confirmed that NMT systems consistently produce more fluent translations than PBMT systems. That manual evaluation only covered language directions into English. In this experiment, we extend that conclusion to language directions out of English.
- Neural LMs consistently outperform *n*-gram based LMs when assessing the fluency of *real* text (Kim et al., 2016; Enarvi and Kurimo, 2016). Thus, we have used the most accurate automatic tool available to measure fluency.

Language direction	Monotone vs.			PBMT vs. Ref.	NMT vs. Ref.
	PBMT	NMT	Ref.		
EN→CS	0.9273	0.9029	0.8295	0.8008	0.7964
EN→DE	0.8006	0.8229	0.7740	0.7560	0.7791
EN→FI	0.8912	0.9172	0.8367	0.7611	0.7819
EN→RO	0.8389	0.8378	0.7937	0.8312	0.8282
EN→RU	0.9342	0.9114	0.8364	0.8249	0.8240
CS→EN	0.7694	0.7589	0.7128	0.8000	0.8015
DE→EN	0.8036	0.7830	0.7409	0.7728	0.7943
RO→EN	0.8693	0.8427	0.8013	0.8187	0.8245
RU→EN	0.8170	0.7891	0.7247	0.7958	0.8069

Table 5: Average Kendall’s tau distance between the word alignments obtained after translating the test set with each MT system being evaluated and a monotone alignment (left); and average Kendall’s tau distance between the word alignments obtained for each MT system’s translation and the word alignments of the reference translation (right). Larger values represent more similar alignments. If the difference between the distances depicted in the two last columns is statistically significant according to paired bootstrap resampling (Koehn, 2004) with $p = 0.05$ and 1 000 iterations, the largest distance is shown in bold.

5 Reordering

In this section we measure the amount of reordering performed by PBMT and NMT systems. Our objective is to empirically determine whether: (i) the recurrent neural networks in NMT systems produce more changes in the word order of a sentence than an PBMT decoder; and whether (ii) these neural networks make the word order of the translations closer to that of the reference.

In order to measure the amount of reordering, we used the *Kendall’s tau distance* between word alignments obtained from pairs of sentences (Birch, 2011, Sec. 5.3.2). As the distance needs to be computed from permutations,⁸ we turned word alignments into permutations by means of the algorithm defined by Birch (2011, Sec. 5.2).

For each language direction, we computed word alignments between the source-language side of the test set and the target-language reference, the PBMT output and the NMT output by means of MGIZA++ (Gao and Vogel, 2008). As the test sets are rather small for word alignment (1 999 to 3 000 sentence pairs depending on the language pair), we append bigger parallel corpora to help ensure accurate word alignments and avoid data sparseness. For languages for which in-domain

(news) parallel training data is available (German and Russian), we append that dataset (News Commentary). For the remaining languages (Finnish and Romanian) we use the whole Europarl corpus.

The amount of reordering performed by each system can be estimated as the distance between the word alignments produced by that system and a monotone word alignment. The similarity between the reorderings produced by each MT system and the reorderings in the reference translation can also be estimated as the distance between the corresponding word alignments. Table 5 shows the value of these distances for the language pairs included in our evaluation. The average over all the sentences in the test set of the distance proposed by Birch (2011) is depicted.

It can be observed that the amount of reordering introduced by both types of MT systems is lower than the quantity of reordering in the reference translation. NMT generally produces more changes in the structure of the sentence than PBMT. This is the case for all language pairs but two (EN→DE and EN→FI). A possible explanation for these two exceptions is the following: in the former language pair, the PBMT system is hierarchical (Williams et al., 2016) while in the latter, the output was reranked with neural LMs.

Concerning the similarity between the reorderings produced by both MT systems and those in the reference translation, out of 9 directions, in 5 directions the NMT system performs a reordering

⁸A permutation between a source-language sentence and a target-language sentence is defined as the set of operations that need to be carried out over the words in the source-language sentence to reflect the order of the words in the target-language sentence (Birch, 2011, Sec. 5.2).

closer to the reference, in 1 direction the PBMT system performs a reordering closer to the reference and in the remaining 3 directions the differences are not statistically significant. That is, NMT generally produces reorderings which are closer to the reference translation. The exceptions to this trend, however, do not exactly correspond to the language pairs for which NMT underperformed PBMT.

In summary, NMT systems achieve, in general, a higher degree of reordering than pure, phrase-based PBMT systems, and, overall, this reordering results in translations whose word order is closer to that of the reference translation.

6 Sentence Length

In this experiment we aim to find out whether the performances of NMT and PBMT are somehow sensitive to sentence length. In this regard, Bentivogli et al. (2016) found that, for transcribed speeches, NMT outperformed PBMT regardless of sentence length while also noted that NMT’s performance degraded faster than PBMT’s as sentence length increases. It should be noted, however, that sentences in our content type, news, are considerably longer than sentences in transcribed speeches.⁹ Hence, the current experiment will determine to what extent the findings on transcribed speeches stand also for texts made of longer sentences.

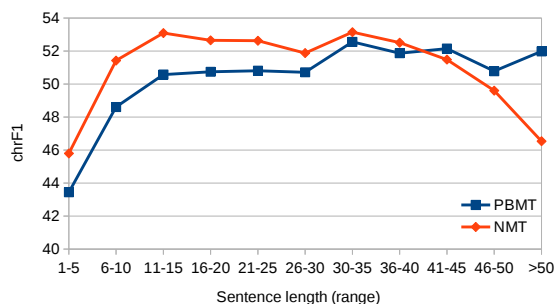


Figure 1: NMT and PBMT chrF1 scores on subsets of different sentence length for the language direction EN→FI.

We split the source side of the test set in subsets of different lengths: 1 to 5 words (1-5), 6 to 10 and so forth up to 46 to 50 and finally longer than 50 words (> 50). We then evaluate the out-

⁹According to Ruiz et al. (2014), sentences of transcribed speeches in English average to 19 words while sentences in news average to 24 words.

puts of the top PBMT and NMT submissions for those subsets with the chrF1 evaluation metric. Figure 1 presents the results for the language direction EN→FI. We can observe that NMT outperforms PBMT up to sentences of length 36-40, while for longer sentences PBMT outperforms NMT, with PBMT’s performance remaining fairly stable while NMT’s clearly decreases with sentence length. The results for the other language directions exhibit similar trends.



Figure 2: Relative improvement of the best NMT versus the best PBMT submission on chrF1 for different sentence lengths, averaged over all the language pairs considered.

Figure 2 shows the relative improvements of NMT over PBMT for each sentence length subset, averaged over all the 9 language directions considered. We observe a clear trend of this value decreasing with sentence length and in fact we found a strong negative Pearson correlation (-0.79) between sentence length and the relative improvement (chrF1) of the best NMT over the best PBMT system.

The correlations for each language direction are shown in Table 6. We observe negative correlations for all the language directions except for DE→EN.

Direction	CS	DE	FI	RO	RU
From EN	-0.72	-0.26	-0.89	-0.01	-0.74
Into EN	-0.19	0.10	-	-0.36	-0.70

Table 6: Pearson correlations between sentence length and relative improvement (chrF1) of the best NMT over the best PBMT system for each language pair.

7 Error Categories

In this experiment we assess the performance of NMT versus PBMT systems on a set of error

Error type	EN→CS	EN→DE	EN→FI	EN→RO	EN→RU	Average
Inflection	-16.18%	-13.26%	-11.65%	-15.13%	-16.79%	-14.60%
Reordering	-7.97%	-21.92%	-12.12%	-15.91%	-6.18%	-12.82%
Lexical	-0.44%	-3.48%	-1.09%	2.17%	-0.09%	-0.59%

Table 7: Relative improvement of NMT versus PBMT for 3 error categories, for language directions out of English.

Error type	CS→EN	DE→EN	RO→EN	RU→EN	Average
Inflection	-4.38%	-2.47%	-3.65%	-21.12%	-7.91%
Reordering	-8.68%	-21.09%	-9.48%	-8.50%	-11.94%
Lexical	-1.92%	-4.91%	-3.90%	5.32%	-1.35%

Table 8: Relative improvement of NMT versus PBMT for 3 error categories, for language directions into English.

categories that correspond to five word-level error classes: inflection errors, reordering errors, missing words, extra words and incorrect lexical choices. These errors are detected automatically using the edit distance, word error rate (WER), precision-based and recall-based position-independent error rates (hPER and rPER, respectively) as implemented in Hjerison (Popović, 2011). These error classes are then defined as follows:

- Inflection error (hINFer). A word for which its full form is marked as a hPER error while its base form matches the base form in the reference.
- Reordering error (hRer). A word that matches the reference but is marked as a WER error.
- Missing word (MISer). A word that occurs as deletion error in WER, is also a rPER error and does not share the base form with any hypothesis error.
- Extra word (EXTer). A word that occurs as insertion error in WER, is also a hPER error and does not share the base form with any reference error.
- Lexical choice error (hLEXer). A word that belongs neither to inflectional errors nor to missing or extra words.

Due to the fact that it is difficult to disambiguate between three of these categories, namely missing words, extra words and lexical choice errors (Popović and Ney, 2011), we group them in

a unique category, which we refer to as lexical errors.

As input, the tool requires the full forms and base forms of the reference translations and MT outputs. For base forms, we use stems for practical reasons. These are produced with the Snowball stemmer from NLTK¹⁰ for all languages except for Czech, which is not supported. For this language we used the aggressive variant in `czech_stemmer`.¹¹

Tables 7 and 8 show the results for language directions out of English and into English, respectively. For all language directions, we observe that NMT results in a notable decrease of both inflection (-14.6% on average for language directions out of EN and -7.91% for language directions into EN) and reordering (-12.82% from EN and -11.94 into EN) errors. The reduction of reordering errors is compatible with the results of the experiment presented in Section 5.¹²

Differences in performance for the remaining error category, lexical errors, are much smaller. In addition, the results for that category show a mixed picture in terms of which paradigm is better, which makes it difficult to derive conclusions that apply regardless of the language pair. Out of En-

¹⁰<http://www.nltk.org>

¹¹http://research.variancia.com/czech_stemmer/

¹²Although the results depicted both in this section and in Section 5 show that NMT performs better reordering in general, results for particular language pairs are not exactly the same in both sections. This is due to the fact that the quality of the reordering is computed in different ways. In this section, only those words that match the reference are considered when identifying reordering errors, while in Section 5 all the words in the sentence are taken into account. That said, in Section 5 the precision of the results depends on the quality of word alignment.

glish, NMT results in slightly less errors (0.59% decrease on average) for all target languages except for RO (2.17% increase). Similarly, in the opposite language direction, NMT also results in slightly better performance overall (1.35% error reduction on average), and looking at individual language directions NMT outperforms PBMT for all of them except RU→EN.

8 Conclusions

We have conducted a multifaceted evaluation to compare NMT versus PBMT outputs across a number of dimensions for 9 language directions. Our aim has been to shed more light on the strengths and weaknesses of the newly introduced NMT paradigm, and to check whether, and to what extent, these generalise to different families of source and target languages. Hereunder we summarise our findings:

- The outputs of NMT systems are considerably different compared to those of PBMT systems. In addition, there is higher inter-system variability in NMT, i.e. outputs by pairs of NMT systems are more different between them than outputs by pairs of PBMT systems.
- NMT outputs are more fluent. We have corroborated the results of the manual evaluation of fluency at WMT16, which was conducted only for language directions into English, and we have shown evidence that this finding is true also for directions out of English.
- NMT systems introduce more changes in word order than pure PBMT systems, but less than hierarchical PBMT systems.¹³ Nevertheless, for most language pairs, including those for which the best PBMT system is hierarchical, NMT's reorderings are closer to the reorderings in the reference than those of PBMT. This corroborates the findings on reordering by Bentivogli et al. (2016).
- We have found negative correlations between sentence length and the improvement brought by NMT over PBMT for the majority of the languages examined. While for most sentence lengths NMT outperforms PBMT, for very long sentences PBMT outperforms

¹³The latter finding applies only to one language direction as only for that one the best PBMT system is hierarchical.

NMT. The latter was not the case in the work by Bentivogli et al. (2016). We believe the reason behind this different finding is twofold. Firstly, the average sentence length in their evaluation dataset was considerably shorter; and secondly, the NMT systems included in our evaluation operate on subword units, which increases the effective sentence length they have to deal with.

- NMT performs better in terms of inflection and reordering consistently across all language directions. We thus confirm that the findings of Bentivogli et al. (2016) regarding these two error types apply to a wide range of language directions. Differences regarding lexical errors are much smaller and inconsistent across language directions; for 7 of them NMT outperforms PBMT while for the remaining 2 the opposite is true.

The results for some of the evaluations, especially error categories (Section 7) have been analysed only superficially, looking at what conclusions can be derived that apply regardless of language direction. Nevertheless, all our data is publicly released,¹⁴ so we encourage interested parties to use this resource to conduct deeper language-specific studies.

Acknowledgments

The research leading to these results is supported by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement PIAP-GA-2012-324414 (Abu-MaTran) and by Science Foundation Ireland through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at Dublin City University.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR 2015*, San Diego, CA, USA.
- Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, TX, USA, November.

¹⁴https://github.com/antot/neural_vs_phrasebased_smt_eacl17

- Alexandra Birch. 2011. *Reordering metrics for statistical machine translation*. Ph.D. thesis, The University of Edinburgh.
- Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August.
- Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the Deep Learning and Representation Learning Workshop, NIPS*.
- Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1693–1703, Berlin, Germany, August.
- Marta R. Costa-Jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation.
- Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June.
- Shuoyang Ding, Kevin Duh, Huda Khayrallah, Philipp Koehn, and Matt Post. 2016. The JHU Machine Translation Systems for WMT 2016. In *Proceedings of the First Conference on Machine Translation*, pages 272–280, Berlin, Germany, August.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.
- Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A Joint Sequence Translation Model with Integrated Reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1045–1054, Portland, Oregon, USA, June.
- Seppo Enarvi and Mikko Kurimo. 2016. TheanoLM – An Extensible Toolkit for Neural Network Language Modeling. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association*.
- Christian Federmann. 2012. Appraise: An open-source toolkit for manual evaluation of machine translation output. *The Prague Bulletin of Mathematical Linguistics*, 98:25–35, September.
- Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing, SETQA-NLP '08*, pages 49–57, Columbus, Ohio.
- Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2016. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, FirstView:1–28, 1.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT. In *Proceedings of the First Conference on Machine Translation*, pages 319–325, Berlin, Germany, August.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749, Phoenix, Arizona, USA, February.
- Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 4, pages 388–395, Barcelona, Spain.
- Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.
- Chi-kiu Lo, Colin Cherry, George Foster, Darlene Stewart, Rabib Islam, Anna Kazantseva, and Roland Kuhn. 2016. NRC Russian-English Machine Translation System for WMT 2016. In *Proceedings of the First Conference on Machine Translation*, pages 326–332, Berlin, Germany, August.
- Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July.

- Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Comput. Linguist.*, 37(4):657–688, December.
- Maja Popović. 2011. Hjerson: An open source tool for automatic error classification of machine translation output. *The Prague Bulletin of Mathematical Linguistics*, 96:59–67.
- Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September.
- Nicholas Ruiz and Marcello Federico. 2014. Complexity of spoken versus written language for machine translation. In *17th Annual Conference of the European Association for Machine Translation, EAMT*, pages 173–180, Dubrovnik, Croatia, June.
- Víctor M. Sánchez-Cartagena and Antonio Toral. 2016. Abu-matran at wmt 2016 translation task: Deep learning, morphological segmentation and tuning on character sequences. In *Proceedings of the First Conference on Machine Translation*, pages 362–370, Berlin, Germany, August.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany, August.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1693–1703, Berlin, Germany, August.
- Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.
- Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, Barry Haddow, and Ondřej Bojar. 2016. Edinburgh’s statistical machine translation systems for wmt16. In *Proceedings of the First Conference on Machine Translation*, pages 399–410, Berlin, Germany, August.

Personalized Machine Translation: Preserving Original Author Traits

Ella Rabinovich

Department of Computer Science
University of Haifa, Israel
& IBM Research - Haifa
ellarabi@gmail.com

Shachar Mirkin

IBM Research - Haifa
Mount Carmel, Haifa
31905, Israel
shacharm@il.ibm.com

Raj Nath Patel

C-DAC Mumbai
Gulmohar Cross Road No. 9, Juhu
Mumbai-400049, India
patelrajnath@gmail.com

Lucia Specia

Department of Computer Science
University of Sheffield, United Kingdom
l.specia@sheffield.ac.uk

Shuly Wintner

Department of Computer Science
University of Haifa, Israel
shuly@cs.haifa.ac.il

Abstract

The language that we produce reflects our personality, and various personal and demographic characteristics can be detected in natural language texts. We focus on one particular personal trait of the author, *gender*, and study how it is manifested in original texts and in translations. We show that author's gender has a powerful, clear signal in original texts, but this signal is obfuscated in human and machine translation. We then propose simple domain-adaptation techniques that help retain the original gender traits in the translation, without harming the quality of the translation, thereby creating more personalized machine translation systems.

1 Introduction

Among many factors that mold the makeup of a text, gender and other authorial traits play a major role in our perception of the content we face. Many studies have shown that these traits can be identified by means of automatic classification methods. Classical examples include gender identification (Koppel et al., 2002), and authorship attribution and profiling (Seroussi et al., 2014). Most research, however, addressed texts in a single language, typically English.

We investigate a related but different question: we are interested in understanding what happens to personality and demographic textual markers during the translation process. It is generally agreed that good translation goes beyond transformation of the original content, by preserving more subtle and implicit characteristics inferred by author's personality, as well as era, geography, and various cultural and sociological aspects. In this work we explore whether translations preserve the

stylistic characteristic of the author and, furthermore, whether the prominent signals of the source are retained in the target language.

As a first step, we focus on *gender* as a demographic trait (partially due to the absence of parallel data annotated for other traits). We evaluate the accuracy of automatic gender classification on original texts, on their manual translations and on their automatic translations generated through statistical machine translation (SMT). We show that while gender has a strong signal in originals, this signal is obfuscated in human and machine translation. Surprisingly, determining gender over manual translation is even harder than over SMT; this may be an artifact of the translation process itself or the human translators involved in it.

Mirkin et al. (2015) were the first to show that authorial gender signals tend to vanish through both manual and automatic translation, using a small TED talks dataset. We use their data and extend it with a version of Europarl that we annotated with age and gender (§3). Furthermore, we conduct experiments with two language pairs, in both directions (§4). We also adopt a different classification methodology based on the finding that the translation process itself has a stronger signal than the author's gender (§4.1).

We then move on to assessing gender traits in SMT (§5). Since SMT systems typically do not take personality or demographic information into account, we hypothesize that the author's style, affected by their personality, will fade. Furthermore, we propose simple domain-adaptation techniques that do consider gender information and can therefore better retain the original traits. We build "gender-aware" SMT systems, and show (§6) that they retain gender markers while preserving general translation quality. Our findings therefore suggest that SMT can be made much more personalized, leading to translations that are more faith-

ful to the style of the original texts.

Finally, we analyze the prominent features that reflect gender in originals and translations (§7). Our experiments reveal that gender markers differ greatly by language, and the specific source language has a significant impact on the features and classification accuracy of the translated text. In particular, gender traits of the original language overshadow those of the target language in both manual and automatic translation products.

The **main contributions** of this paper are thus: (i) a new parallel corpus annotated with gender and age information, (ii) an in-depth assessment of the projection of gender traits in manual and automatic translation, and (iii) experiments showing that gender-personalized SMT systems better project gender traits while maintaining translation quality.

2 Related work

While modeling of demographic traits has been proven beneficial in some NLP tasks such as sentiment analysis (Volkova et al., 2013) or topic classification (Hovy, 2015), very little attention has been paid to translation. We provide here a brief summary of research relevant to our work.

Machine translation (MT) Virtually no previous work in MT takes into account personal traits. State-of-the-art MT systems are built from examples of translations, where the general assumption is that the more data available to train models, the better, and a single model is usually produced. Exceptions to this assumption revolve around work on domain adaption, where systems are customized by using data that comes from a particular text domain (Hasler et al., 2014; Cuong and Sima'an, 2015); and work on data cleaning, where spurious data is removed from the training set to ensure the quality of the final models (Cui et al., 2013; Simard, 2014). Personal traits, sometimes well marked in the translation examples, are therefore not explicitly addressed. Learning from different, sometimes conflicting writing styles can hinder model performance and lead to translations that are unfaithful to the source text.

Focusing on reader preferences, Mirkin and Meunier (2015) used a collaborative filtering approach from recommender systems, where a user's preferred translation is predicted based on the preferences of similar users. However, the user preferences in this case refer to the overall choice

between MT systems of a specific reader, rather than a choice based on traits of the writer. Mirkin et al. (2015) motivated the need for personalization of MT models by showing that automatic translation does not preserve demographic and psychometric traits. They suggested treating the problem as a domain adaptation one, but did not provide experimental results of personalized MT models.

Gender classification A large body of research has been devoted to isolating distinguishing traits of male and female linguistic variations, both theoretically and empirically. Apart from content, male and female speech has been shown to exhibit stylistic and syntactic differences. Several studies demonstrated that literary texts and blog posts produced by male and female writers can be distinguished by means of automatic classification, using (content-independent) function words and n-grams of POS tags (Koppel et al., 2002; Schler et al., 2006; Burger et al., 2011).

Although the tendencies of *individual word* usage are a subject of controversy, distributions of *word categories* across male and female English speech is nearly consensual: pronouns and verbs are more frequent in female texts, while nouns and numerals are more typical to male productions. Newman et al. (2008) carried out a comprehensive empirical study corroborating these findings with large and diverse datasets.

However, little effort has been dedicated to investigating the variation of individual markers of demographic traits across different languages. Johannsen et al. (2015) conducted a large-scale study on linguistic variation over age and gender across multiple languages in a social media domain. They showed that gender differences captured by shallow syntactic features were preserved across languages, when examined by linguistic categories. However, they did not study the distribution of individual gender markers across domains and languages. Our work demonstrates that while marker categories are potentially preserved, individual words typical to male and female language vary across languages and, more prominently, across different domains.

Authorial traits in translationese A large body of previous research has established that translations constitute an autonomic *language variety*: a special dialect of the target language, often re-

ferred to as *translationese* (Gellerstam, 1986). Recent corpus-based investigations of translationese demonstrated that originals and translations are distinguishable by means of supervised and unsupervised classification (Baroni and Bernardini, 2006; Volansky et al., 2015; Rabinovich and Wintner, 2015). The identification of machine-translated text has also been proven an easy task (Arase and Zhou, 2013; Aharoni et al., 2014).

Previous work has investigated how gender artifacts are carried over into human translation in the context of social and gender studies, as well as cultural transfer (Simon, 2003; Von Flotow, 2010). Shlesinger et al. (2009) conducted a computational study exploring the implications of the translator’s gender on the final product. They conclude that “the computer could not be trained to accurately predict the gender of the translator”. Preservation of authorial style in literary translations was studied by Lynch (2014), identifying Russian authors of translated English literature, by using (shallow) stylistic and syntactic features. Forsyth and Lam (2014) investigated authorial discriminability in translations of French originals into English, inspecting two distinct human translations, as well as automatic translation of the same sources.

Our work, to the best of our knowledge, is the first to automatically identify speaker gender in manual, and more prominently, automatic translations over multiple domains and language-pairs, examining distribution of gender markers in source and target languages.

3 Europarl with demographic info

We created a resource¹ based on the parallel corpus of the European Parliament (Europarl) Proceedings (Koehn, 2005). More specifically, we utilize the extension of its *en-fr* and *en-de* parallel versions (Rabinovich et al., 2015), where each sentence-pair is annotated with speaker name, the original language the sentence was uttered in, and the date of the corresponding session protocol. To extend speaker information with demographic properties, we used the Europarl website’s MEP information pages² and applied a procedure of gender and age identification, as further detailed in §3.1.

The final resource comprises *en-fr* and *en-de* parallel bilingual corpora where metadata of mem-

bers of the European Parliament (MEPs) is enriched with their gender and age at the time of the corresponding session. The data is restricted to sentence-pairs originally produced in English, French, or German. Table 1 provides statistics on the two datasets. We also release the full list of 3,586 MEPs with their meta information.

	<i>en-fr</i>	<i>fr-en</i>	<i>en-de</i>	<i>de-en</i>
male	100K	67K	101K	88K
female	44K	40K	61K	43K
total	144K	107K	162K	131K

Table 1: Europarl corpora (EP) statistics (# of sentence-pairs); gender refers to an author of the source utterance.

3.1 Identification of MEP gender

Gender annotation was conducted using three different resources: Wikidata, Genderize and AlchemyVision, which we briefly describe below.

Wikidata (Vrandečić and Krötzsch, 2014) is a human-curated knowledge repository of structured data from Wikipedia and other Wikimedia projects. Wikidata provides an API³ through which one can retrieve details about people in the repository, including place and date of birth, occupation, and gender. For MEPs found in the Wikidata, we first verified that the person holds (or held) a position of Member of the European Parliament and if so, retrieved the gender. Wikidata information is not complete: not all MEP names, positions or gender data is included. In total we obtained gender information for 2,618 MEPs (73% of the total 3,586), of which 1,882 (72%) are male and 736 female (28%).

Genderize⁴ is an open resource containing over 2 million distinct names grouped by countries. It determines people’s gender based on their first name and the country of origin. Provided with the first name and the country a MEP represents.⁵ Genderize was able to predict the gender of 2,785 MEPs, the vast majority of them with a probability of 0.9 or higher. We filtered out the 55 lower-confidence entries, keeping 2,730 MEPs (76% of total), of which 2001 (73%) are male and 729 (27%) female.

³<https://www.mediawiki.org/wiki/Wikibase/API>

⁴<https://genderize.io/>

⁵We assume that the country MEPs represent is highly correlated, if not strictly identical, to their country of origin.

¹Available at <http://cl.haifa.ac.il/projects/pmt>

²<http://www.europarl.europa.eu/meps/en/>

AlchemyVision The European Parliament website maintains a page for every MEP, including personal photos. We classified MEP personal images using AlchemyVision,⁶ a publicly available image recognition service. In total, we retrieved the gender of 2,236 MEPs using AlchemyVision. Similarly to Genderize, we filtered out all predictions with a confidence score below 0.9, thus obtaining the gender of 2,138 MEPs (60% of total), of which 1,528 are male and 610 female (71% and 29%, respectively).

3.2 Resource evaluation and statistics

Even though Wikidata was created manually, to verify its correctness, we manually annotated the gender of 100 randomly selected MEPs with available Wikidata gender information; we found the metadata perfectly accurate. We therefore rely on Wikidata as a gold-standard against which we can assess the accuracy of the two other resources. Table 2 presents the accuracy and coverage of each resource based on this methodology.

resource	Wikidata	Genderize	Alchemy
coverage	73.0	76.1	59.6
accuracy	100.0	99.6	99.1

Table 2: Gender prediction performance (%).

Given information obtained from the three resources, we assign each MEP with a single gender prediction in the following way: whenever it is found in Wikidata (2,618 MEPs), the gender is determined by this resource. Otherwise, if both Genderize and AlchemyVision produced agreed-upon gender information (336 out of 338 cases), we set gender according to this prediction; the same applies to the case where only one of Genderize or AlchemyVision provided a prediction (346 and 178, respectively). We ended up with gender annotation for a total of 3,478 out of 3,586 members. The remaining 108 MEPs (92 male, 16 female) were annotated manually, a rather labor-intensive annotation in this case.

In total, the resource includes 947 (26%) female and 2,639 (74%) male MEPs. Based on the above accuracy estimations, and assuming that manual annotation is correct, the overall accuracy of gender information in this resource is 99.88%.

Utilizing the information on session dates and

⁶<https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/alchemy-vision.html>

MEPs dates of birth available in the metadata, we also annotated each sentence-pair with the age of the MEP at the time the sentence was uttered. To summarize, we release the following resources: (i) meta information for 3,586 MEPs, as described above, (ii) bilingual parallel *en-fr* and *en-de* corpora, where each sentence-pair metadata is enriched with speaker MEPID, gender and age.

4 Experimental setup

We evaluate the extent to which gender traits are preserved in translation by evaluating the accuracy of gender classification of original and translated texts. The rationale is that the more prominent gender markers are in the text, the easier it is to classify the gender of its author.

4.1 Translationese vs. gender traits

Since we use the accuracy of gender identification as our evaluation metric, we isolate the dimension of gender in our data: the classification experiments are carried out separately on original, human translated text, as well as on each one of the MT products. Human, and more prominently, machine translations constitute distinct and distinguishable language variation, characterized by unique feature distributions (§2). We posit that in both human and machine translation products, the differences between original texts and translations overshadow the differences in gender. We corroborate this assumption by analysing a sample data distribution by two dimensions: (i) translation status and (ii) gender. Figure 1 presents the results for the English Europarl corpus. Both charts display data distributions of the same four classes: original (O) and translated (T) English⁷ by male (M) and female (F) speakers (OM, OF, TM, TF). For the sake of visualization, the dimension of function words feature vectors was reduced to 2, using principal component analysis (Jolliffe, 2002). The left graph depicts color-separation by gender (male vs. female), while the right one by translation status (original vs. translated). Evidently, the linguistic variable of translationese stands out against the weaker signal of gender.

4.2 Datasets

In addition to the Europarl corpus annotated for gender (§3), we experimented with a corpus of

⁷This experiment refers to English translated from French; other language-pairs exhibited similar trends.

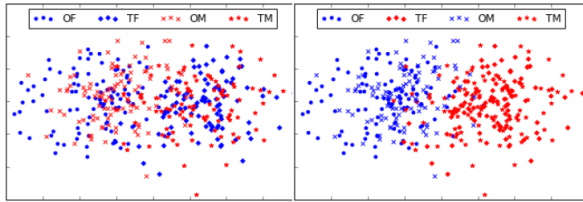


Figure 1: English EP data distributions across two dimensions: gender (left) and trans. status (right).

TED talks (transcripts and translations): a collection of texts from a completely different genre, where demographic traits may manifest differently. Testing the potential benefits of personalized SMT models on these two very diverse datasets allows us to examine the robustness of our approach. We used the TED gender-annotated data from Mirkin et al. (2015).⁸ This corpus contains annotation of the speaker’s gender included in the English-French corpus of the IWSLT 2014 Evaluation Campaign’s MT track (Cettolo et al., 2012). We annotated 68 additional talks from the development and test sets of IWSLT 2014, 2015 and 2016. Using the full set, we split the TED parallel corpora by gender to obtain sub-corpora of 140K and 43K sentence pairs for male and female speakers, respectively.

The sizes of the datasets used for training, tuning and testing of SMT models are shown in Table 3. Relatively large test sets are used for evaluation of the MT results for the sake of reliable per-outcome gender classification (§4.1).

Although the size of the training/tuning/test sets in either direction for any language-pair is the same, their content is different. We use data in both translation directions (i.e., *en-fr* and *fr-en*, or *en-de* and *de-en*) for both SMT experiments. Out of these data, 2K and 15K sentence-pairs (for each gender) are held out for tuning and test, respectively, where they comply with the translation direction. That is, for *en-fr* experiments, tuning and test sets are sampled from the *en-fr* direction only and vice-versa. The additional bilingual data (ADD) for training the models comes from the gender-unannotated portion of Europarl (all but the gender-annotated sub-corpus detailed in §3) for the EP experiments, and from combining TED’s male and female data for the experiments with TED.

⁸Downloaded from <http://cm.xrce.xerox.com/>.

4.3 Classification setting

All datasets were split by sentence, filtering out sentence alignments other than one-to-one. For POS tagging, we employed the Stanford implementation⁹ with its models for English, French and German. We divided all datasets into chunks of approximately 1,000 tokens, respecting sentence boundaries, and normalized the values of lexical features by the actual number of tokens in each chunk. For classification, we used Platt’s sequential minimal optimization algorithm (Keerthi et al., 2001) to train support vector machine classifiers with the default linear kernel (Hall et al., 2009). In all experiments we used (the maximal) equal amount of data from each category (M and F), specifically, 370 chunks for each gender.

Aiming to abstract away from content and capture instead stylistic and syntactic characteristics, we used as our feature set the combination of function words (FW)¹⁰ and (the top-1,000 most frequent) POS-trigrams. We employ 10-fold cross-validation for evaluation of classification accuracy.

4.4 SMT setting

We trained phrase-based SMT models with Moses (Koehn et al., 2007), an open source SMT system. KenLM (Heafield, 2011) was used for language modeling. We trained 5-gram language models with Kneser-Ney smoothing (Chen and Goodman, 1996). The models were tuned using Minimum Error Rate Tuning (MERT) (Och, 2003). Our preprocessing included cleaning (removal of empty, long and misaligned sentences), tokenization and punctuation normalization. The Stanford tokenizer (Manning et al., 2014) was used for tokenization and standard Moses scripts were used for other preprocessing tasks. We used BLEU (Papineni et al., 2002) to evaluate MT quality against one reference translation.

5 Personalized SMT models

In order to investigate and improve gender traits transfer in MT, we devise and experiment with gender-aware SMT models. We demonstrate that despite their simplicity, these models lead to better preservation of gender traits, while not harming the general quality of the translations.

⁹<http://nlp.stanford.edu/software/tagger.shtml>

¹⁰We used the lists of function words available at <https://code.google.com/archive/p/stop-words>.

dataset	language-pair	training			tuning		test	
		M	F	ADD	M	F	M	F
EP	<i>en-fr</i> & <i>fr-en</i>	144K	65K	1.71M	2K	2K	15K	15K
	<i>en-de</i> & <i>de-en</i>	170K	86K	1.50M	2K	2K	15K	15K
TED	<i>en-fr</i>	117K	21K	138K	2K	2K	20K	20K

Table 3: MT datasets split for train, tuning and test, after cleaning.

We treat the task of personalizing SMT models as a domain adaptation task, where the *domain* is the gender. We applied two common techniques: (i) gender-specific model components (phrase table and language model (LM)) and (ii) gender-specific tuning sets. These personalized configurations are further compared to a baseline model where gender information is disregarded, as described below. In all cases, we use a single re-ordering table built from the entire training set.

Baseline The baseline (*MT-B*) system was trained using the complete parallel corpus available for a language-pair. The training set contained both gender-specific and unannotated data, but no distinction was made between them. A single translation model and a single LM were built, and the model was tuned using a random sample of 2K sentence-pairs from the mixed data dedicated for tuning, preserving, therefore, the gender distribution of the underlying dataset.

Personalized models These models use three datasets: male, female, and additional in-domain bilingual data. Two configurations were devised: *MT-P1*, a model with three phrase tables and three LMs trained on the three datasets; and *MT-P2*, where for each gender a phrase table and a language model were built using only the gender-specific data, as well as a general phrase table and LM. In both configurations, each of the two genderized model variants was tuned using the gender-specific tuning set. In order to evaluate the translation quality of a personalized model, we separately translated the male and female source segments, merged the outputs and evaluated the merged result.

6 Results

Recall that we use the accuracy of gender classification as a measure of the strength of gender markers in texts. We assessed this accuracy below on originals and (human and machine) translations. First, however, we establish that the quality of SMT is not harmed with our personalized

models.

MT evaluation We trained a baseline (*MT-B*) and two personalized models (*MT-P1* and *MT-P2*) for each language pair as detailed in §5. The BLEU scores of *en-fr* and *fr-en* personalized models were 38.42, 38.34 and 37.16, 37.16, with the baseline models scoring 38.65 and 37.35, respectively. Similarly, for experiments with *en-de* and *de-en* and the TED data, the baseline scores (21.95, 26.37 and 33.25) were only marginally higher than those of the personalized models (21.65, 21.80; 26.35, 26.21; and 33.19, 33.16), with differences ranging from 0.02 to 0.3. Neither *MT-P1* nor *MT-P2* was consistently better than the other. We conclude, therefore, that all MT systems are comparable in terms of general quality.

Classification accuracy Tables 4 and 5 present the results of gender classification accuracy in original (O), human- (HT) and machine-translated texts in the EP corpus. Female texts are distinguishable from their male counterparts with 77.3% and 77.1% accuracy for English originals, in line with accuracies reported in the literature (Koppel et al., 2002). Classification of original French and German texts reach 81.4% (Table 4) and 76.1% (Table 5), respectively.

dataset	precision		recall		acc.
	M	F	M	F	
<i>en</i> O	77.7	76.9	76.5	78.1	77.3
<i>fr</i> O	80.9	81.9	82.2	80.5	81.4
<i>fr-en</i> HT	75.6	74.4	73.8	76.2	75.0
<i>fr-en</i> <i>MT-B</i>	77.0	78.2	78.6	76.5	77.6
<i>fr-en</i> <i>MT-P1</i>	82.0	80.7	80.3	82.4	81.4
<i>fr-en</i> <i>MT-P2</i>	79.1	81.0	81.6	78.4	80.0
<i>en-fr</i> HT	56.6	56.4	55.7	57.3	56.5
<i>en-fr</i> <i>MT-B</i>	60.2	60.1	60.0	60.3	60.1
<i>en-fr</i> <i>MT-P1</i>	62.7	63.0	63.5	62.2	62.8
<i>en-fr</i> <i>MT-P2</i>	65.2	65.3	65.4	65.1	65.3

Table 4: EP *en-fr*, *fr-en* classification scores (%).

Evidently, gender traits are significantly obfuscated by both manual and non-personalized ma-

dataset	precision		recall		acc.
	M	F	M	F	
<i>en O</i>	77.5	76.7	76.5	77.7	77.1
<i>de O</i>	76.4	75.7	75.4	76.8	76.1
<i>de-en HT</i>	68.6	67.9	67.3	69.2	68.2
<i>de-en MT-B</i>	69.3	69.9	70.3	68.9	69.6
<i>de-en MT-P1</i>	77.4	75.9	75.1	78.1	76.6
<i>de-en MT-P2</i>	76.2	75.7	75.4	76.5	75.9
<i>en-de HT</i>	59.8	59.7	59.5	60.0	59.7
<i>en-de MT-B</i>	63.8	64.0	64.3	63.5	63.9
<i>en-de MT-P1</i>	69.6	69.4	69.2	69.7	69.5
<i>en-de MT-P2</i>	66.7	67.7	68.6	65.7	67.2

Table 5: EP *en-de*, *de-en* classification scores (%).

chine translation. The relatively low accuracy for human translation can be (partially) explained by the extensive editing procedure applied on Europarl proceedings prior to publishing (Cucchi, 2012), as well as the potential “fingerprints” of (male or female) human translators left on the final product.

Both *MT-P1* and *MT-P2* models yield translations that better preserve gender traits, compared to their manual and gender-agnostic automatic counterparts: accuracy improvements vary between 3.8 for *fr-en* translations to 7.0 percent points for *de-en*¹¹ (*MT-P1* vs *MT-B* in both cases). Per-class precision and recall scores do not exhibit significant differences, despite the unbalanced amount of per-gender data used for training the MT models.

Gender classification results in the TED dataset are presented in Table 6. The classification accuracy of English originals is 80.4%. While, similarly to Europarl, the gender signal is generally weakened in human translations¹² and baseline MT, overall accuracies are in most cases higher than in Europarl across all models. We attribute this difference to the more emotional and personal nature of TED speeches, compared with the formal language of the EP proceedings. Both personalized SMT models significantly outperform their baseline counterpart, as well as the manual translation, yielding 77.2% and 77.7% accuracy for *MT-P1* and *MT-P2*, respectively.

¹¹All differences between *MT-P1* and *MT-P2* and baseline models are statistically significant.

¹²TED talks are subtitled, rather than transcribed, undergoing some editing and rephrasing.

dataset	precision		recall		acc.
	M	F	M	F	
<i>en O</i>	81.2	79.7	79.2	81.6	80.4
<i>en-fr HT</i>	74.0	73.5	73.2	74.3	73.8
<i>en-fr MT-B</i>	71.3	70.1	69.2	72.2	70.7
<i>en-fr MT-P1</i>	77.5	76.8	76.5	77.8	77.2
<i>en-fr MT-P2</i>	78.2	77.2	76.8	78.6	77.7

Table 6: TED *en-fr* classification scores (%).

7 Analysis

Analysis of gender markers To analyze the extent to which personal traits are preserved in translations, we extract the set of most discriminative FWs in various texts by employing the InfoGain feature selection procedure (Gray, 1990). Gender markers vary across *original* languages (with few exceptions); in EP, the most discriminating English features are *also*, *very*, *perhaps*, *as*, *its*, *others*, *you*. The French list includes *on*, *vous*, *dire*, *afin*, *doivent*, *doit*, *aussi*, *avait*, *voilà*, *je*, while the German list consists of *wir*, *man*, *wirklich*, *sollten*, *von*, *für*, *dass*, *allen*, *ob*. The list of discriminative markers in the TED English dataset contains mainly personal pronouns: *she*, *her*, *I*, *you*, *my*, *our*, *me*, *and*, *who*, *it*.

Figure 2 (top) presents weights assigned to various gender markers by the InfoGain attribute evaluator in originals and translations. Gender markers are carried over to (both manual and machine) translations to an extent that overshadows the original markers of the target language. In particular, the markers observed in translated English mirror their original French counterparts, in the same marker role: *I* (M) in English translations reflecting the original French *je* (M), *say* (M) reflecting *dire* (M), *must* (F) translated from *doit* (F) and *doivent* (F); the latter contradicting the original English *must* which characterizes M speech. The original English prominent gender markers (e.g., *also*, *very*) almost completely lose their discriminative power in translations. A similar phenomenon is exhibited by English translations from German, as depicted in Figure 2 (bottom): the German *wir* (*we*), *für* (*for*) and *ob* (*whether*) are preserved in (both manual and machine) English translations, in the same marker role.

We conclude that (i) gender traits in translation are weakened, compared to their originals. Furthermore, (ii) translations tend to embrace gender tendencies of the original language, thus resulting

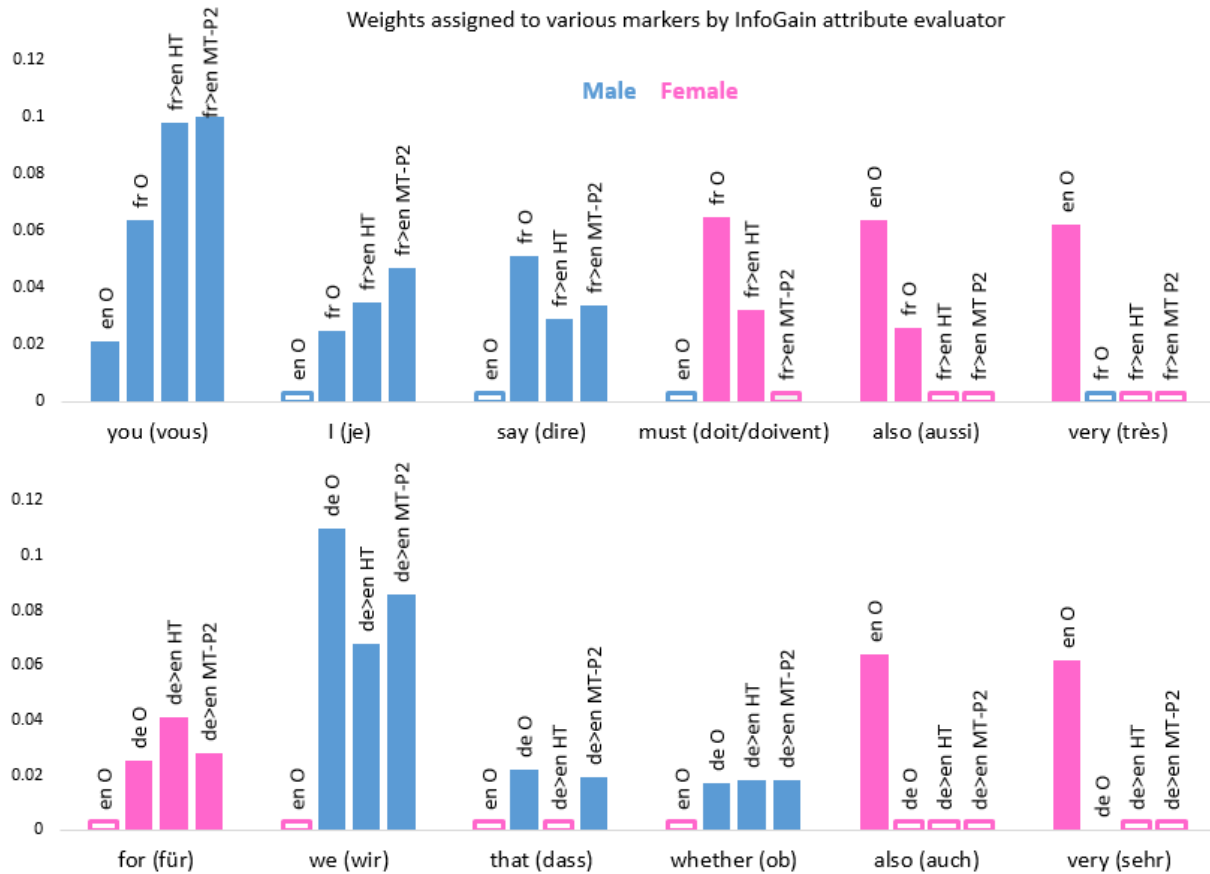


Figure 2: Persistence of *en* and *fr* markers in *fr-en* translations (top); *en* and *de* markers in *de-en* translations (bottom). The transparent bars refer to (weak) F/M markers, assigned weight < 0.01 by InfoGain.

in a *hybrid* outcome, where male and female traits are affected both by markers of the source and (to a much lesser extent) the target language.

Capturing the “personalization” effect Both manual- and all machine-translations of Europarl are tested on a strictly identical set of sentences; therefore, the performance gap introduced by personalized SMT models can be captured by a subset of sentences misclassified by the baseline model, but classified correctly when applying a more personalized approach. The inspection of differences in these translations can shed some light on the underlying nature of our personalized models. Table 7 (top) shows manual, baseline, and personalized machine translations of examples of French and German sentences. The translation of the French word “vraiment” (in a *male* utterance) varies in English as “really” or “exactly”, where the former is more frequent in female English texts, and the latter is a male marker. The choice of a *male* English marker over its *female* equivalent by the gender-aware SMT model demonstrates the

effect of personalization as proposed in this paper. The translations of the German *female* sentence into English, as presented in Table 7 (bottom), further highlight this phenomenon by choosing the English *female* marker *think* in its personalized translation over the more neutral *consider* and *believe* in the manual and baseline versions, respectively.

8 Conclusions

We presented preliminary results of employing personalized SMT models for better preservation of gender traits in automatic translation. This work leaves much room for further research and practical activities. Authors’ personal traits are utilized by recommendation systems, conversational agents and other personalized applications. While resources annotated for personality traits mainly exist for English (and recently, for a small set of additional languages), they are scarce or missing from most other languages. Employing MT models that are sensitive to authors’ personal traits can

<i>fr O</i>	... on a corrigé la traduction du mot qui a été traduit en français par “propriété” qui n’est pas vraiment la même chose qu’ “appropriation”.
<i>fr-en HT</i>	... it had been translated into French using the word for “property”, which is not really the same thing as “ownership”.
<i>fr-en MT-B</i>	... it was corrected the translation of the word which has been translated into French as “ownership”, which is not really the same as “ownership”.
<i>fr-en MT-P1</i>	... it has corrected the translation of the word which has been translated into French as “ownership”, which is not exactly the same as “ownership”.
<i>de O</i>	Entsprechend halte ich es auch für notwendig , daß die Kennzeichnung möglichst schnell und verpflichtend eingeführt wird, und zwar für Rinder und für Rindfleisch .
<i>de-en HT</i>	Accordingly, I consider it essential that both the identification of cattle and the labelling of beef be introduced as quickly as possible on a compulsory basis.
<i>de-en MT-B</i>	Similarly, I believe that it is necessary , as quickly as possible and that compulsory labelling will be introduced, and for bovine animals and for beef and veal.
<i>de-en MT-P1</i>	Accordingly, I also think it is essential that the labelling and become mandatory as quickly as possible, and for bovine animals and for beef.

Table 7: Translation of *fr* (M) and *de* (F) sentences into English manually, and by different MT models.

facilitate user modeling in other languages as well as augment English data with translated content.

Our future plans include experimenting with more sophisticated MT models, and with additional demographic traits, domains and language-pairs.

Acknowledgments

This research was partly supported by the H2020 QT21 project (645452, Lucia Specia). We are grateful to Sergiu Nisoi for sharing the initial collection of properties of Members of the European Parliament. We also thank our anonymous reviewers for their constructive feedback.

References

- Roei Aharoni, Moshe Koppel, and Yoav Goldberg. 2014. Automatic detection of machine translated text and translation quality estimation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 289–295.
- Yuki Arase and Ming Zhou. 2013. Machine translation detection from monolingual web-text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1597–1607.
- Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of Translationese: Machine-learning the difference between original and trans-

lated text. *Literary and Linguistic Computing*, 21(3):259–274, September.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP ’11*, pages 1301–1309, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit³: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*.

Costanza Cucchi. 2012. Dialogic features in EU non-native parliamentary debates. *Review of the Air Force Academy*, 11(3):5–14.

Lei Cui, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Bilingual data cleaning for smt using graph-based random walk. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 340–345, Sofia, Bulgaria.

Hoang Cuong and Khalil Sima’an. 2015. Latent domain word alignment for heterogeneous corpora. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 398–408, Denver, USA.

Richard S Forsyth and Phoenix WY Lam. 2014. Found in translation: To what extent is authorial discriminability preserved by translators? *Literary and Linguistic Computing*, 29(2):199–217.

- Martin Gellerstam. 1986. Translationese in Swedish novels translated from English. In Lars Wollin and Hans Lindquist, editors, *Translation Studies in Scandinavia*, pages 88–95. CWK Gleerup, Lund.
- Robert M Gray. 1990. Entropy and information. In *Entropy and Information Theory*, pages 21–55. Springer.
- Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.
- Eva Hasler, Barry Haddow, and Philipp Koehn. 2014. Dynamic topic adaptation for smt using distributional profiles. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 445–456, Baltimore, USA.
- Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.
- Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 752–762, Beijing, China.
- Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 103–112, Beijing, China, July. Association for Computational Linguistics.
- Ian Jolliffe. 2002. *Principal component analysis*. Wiley Online Library.
- S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. 2001. Improvements to platt’s smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL ’07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. MT Summit.
- Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *LLC*, 17(4):401–412.
- Gerard Lynch. 2014. A supervised learning approach towards profiling the preservation of authorial style in literary translations. In *COLING*, pages 376–386.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.
- Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2019–2025, Lisbon, Portugal, September. Association for Computational Linguistics.
- Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1102–1108, Lisbon, Portugal, September. Association for Computational Linguistics.
- Matthew L. Newman, Carla J. Groom, Lori D. Handelman, and James W. Pennebaker. 2008. Gender differences in language use: An analysis of 14,000 text samples. *Discourse Processes*, 45(3):211–236.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1 (ACL 2003)*, ACL ’03, pages 160–167.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, Philadelphia, Pennsylvania, USA.
- Ella Rabinovich and Shuly Wintner. 2015. Unsupervised identification of translationese. *Transactions of the Association for Computational Linguistics*, 3:419–432.
- Ella Rabinovich, Shuly Wintner, and Ofek Luis Lewinsohn. 2015. The haifa corpus of translationese. *arXiv preprint arXiv:1509.03611*.
- Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2006. Effects of age and gender on blogging. In *Computational Approaches to Analyzing Weblogs, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-03, Stanford, California, USA, March 27-29, 2006*, pages 199–205.
- Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship attribution with topic models. *Comput. Linguist.*, 40(2):269–310.

- Miriam Shlesinger, Moshe Koppel, Noam Ordan, and Brenda Malkiel. 2009. Markers of translator gender: do they really matter? *Internet. Disponível em [http://u. cs. biu. ac. il/~ koppel/Publications. ht ml](http://u.cs.biu.ac.il/~koppel/Publications.ht ml) (consultado em 31 de março de 2011).*
- Michel Simard. 2014. Clean data for training statistical mt: the case of mt contamination. In *Proceedings of the Eleventh Conference of the Association for Machine Translation in the Americas*, pages 69–82, Vancouver, BC, Canada.
- Sherry Simon. 2003. *Gender in translation*. Routledge.
- Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities*, 30(1):98–118, April.
- Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1815–1827, Seattle, Washington, USA.
- Luise Von Flotow. 2010. Gender in translation. *Handbook of Translation Studies*, 1:129–133.
- Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September.

Bilingual Lexicon Induction by Learning to Combine Word-Level and Character-Level Representations

Geert Heyman¹, Ivan Vulić², and Marie-Francine Moens¹

¹LIIR, Department of Computer Science, KU Leuven

²Language Technology Lab, DTAL, University of Cambridge
{geert.heyman, sien.moens}@cs.kuleuven.be
iv250@cam.ac.uk

Abstract

We study the problem of bilingual lexicon induction (BLI) in a setting where some translation resources are available, but unknown translations are sought for certain, possibly domain-specific terminology. We frame BLI as a classification problem for which we design a neural network based classification architecture composed of recurrent long short-term memory and deep feed forward networks. The results show that word- and character-level representations each improve state-of-the-art results for BLI, and the best results are obtained by exploiting the synergy between these word- and character-level representations in the classification model.

1 Introduction

Bilingual lexicon induction (BLI) is the task of finding words that share a common meaning across different languages. Automatically induced bilingual lexicons support a variety of tasks in information retrieval and natural language processing, including cross-lingual information retrieval (Lavrenko et al., 2002; Levow et al., 2005; Vulić and Moens, 2015; Mitra et al., 2016), statistical machine translation (Och and Ney, 2003; Zou et al., 2013), or cross-lingual entity linking (Tsai and Roth, 2016). In addition, they serve as a natural bridge for cross-lingual annotation and model transfer from resource-rich to resource-impooverished languages, finding their application in downstream tasks such as cross-lingual POS tagging (Yarowsky and Ngai, 2001; Täckström et al., 2013; Zhang et al., 2016), dependency parsing (Zhao et al., 2009; Durrett et al., 2012; Upadhyay et al., 2016), semantic role labeling (Padó and Lapata, 2009; van der Plas et al., 2011), to name only a few.

Current state-of-the-art BLI results are obtained by cross-lingual word embeddings (Mikolov et al., 2013b; Faruqui and Dyer, 2014; Gouws et al., 2015; Vulić and Moens, 2016; Duong et al., 2016, inter alia). They significantly outperform traditional count-based baselines (Gaussier et al., 2004; Tamura et al., 2012). Although cross-lingual word embedding models differ on the basis of a bilingual signal from parallel, comparable or monolingual data used in training (e.g., word, sentence, document alignments, translation pairs from a seed lexicon),¹ they all induce word translations in the same manner. (1) They learn a *shared bilingual semantic space* in which all source language and target language words are represented as dense real-valued vectors. The shared space enables words from both languages to be represented in a uniform language-independent manner such that similar words (regardless of the actual language) have similar representations. (2) Cross-lingual semantic similarity between words w and v is then computed as $SF(\vec{w}, \vec{v})$, where \vec{w} and \vec{v} are word representations in the shared space, and SF denotes a similarity function operating in the space (cosine similarity is typically used). A target language word v with the highest similarity score $\arg \max_v SF(\vec{w}, \vec{v})$ is then taken as the correct translation of a source language word w .

In this work, we detect two major gaps in current representation learning for BLI. First, the standard embedding-based approach to BLI learns representations solely on the basis of word-level information. While early BLI works already established that character-level orthographic features may serve as useful evidence for identifying translations (Melamed, 1995; Koehn and Knight, 2002;

¹See recent comparative studies on cross-lingual word embedding learning (Upadhyay et al., 2016; Vulić and Korhonen, 2016) for an in-depth discussion of the differences in modeling and bilingual supervision.

Haghighi et al., 2008), there has been no attempt to learn character-level bilingual representations automatically from the data and apply them to improve on the BLI task. Moreover, while prior work typically relies on simple orthographic distance measures such as edit distance (Navarro, 2001), we show that such character-level representations can be induced from the data. Second, Irvine and Callison-Burch (2013; 2016) demonstrated that bilingual lexicon induction may be framed as a classification task where multiple heterogeneous translation clues/features may be easily combined. Yet, all current BLI models still rely on straightforward similarity computations in the shared bilingual word-level semantic space (see Sect. 2).

Motivated by these insights, we propose a *novel bilingual lexicon induction (BLI) model* that combines automatically extracted word-level and character-level representations in a classification framework. As the seminal bilingual representation model of Mikolov et al. (2013b), our bilingual model learns from a set of training translation pairs, but we demonstrate that the synergy between word-level and character-level features combined within a deep neural network based classification framework leads to improved BLI results when evaluated in the medical domain. BLI has a large value in finding translation pairs in specialized domains such as the medical domain, where general translation resources are often insufficient to capture translations of all domain terminology.

This paper has several contributions:

- (C1)** *On the word level*, we show that framing BLI as a classification problem, that is, using word embeddings as features for classification leads to improved results compared to standard embedding-based BLI approaches (Mikolov et al., 2013b; Vulić and Korhonen, 2016) which rely on similarity metrics in a bilingual semantic space.
- (C2)** *On the character level*, we find that learning character-level representations with an RNN architecture significantly improves results over standard distance metrics used in previous BLI research to operationalize orthographic similarity.
- (C3)** We finally show that it is possible to effectively *combine word- and character-level signals* using a deep feed-forward neural network. The combined model outperforms “single” word-level and character-level BLI models which rely on only one set of features.

2 Background

Word-Level Information for BLI Bilingual lexicon induction is traditionally based on word-level features, aiming at quantifying cross-lingual word similarity on the basis of either (1) context vectors, or (2) automatically induced bilingual word representations. A typical context-vector approach (Rapp, 1995; Fung and Yee, 1998; Gaussier et al., 2004; Laroche and Langlais, 2010; Vulić and Moens, 2013b; Kontonatsios et al., 2014, inter alia) constructs context vectors in two languages using weighted co-occurrence patterns with other words, and a bilingual seed dictionary is then used to translate the vectors. Second-order BLI approaches which represent a word by its monolingual semantic similarity with other words were also proposed, e.g., (Koehn and Knight, 2002; Vulić and Moens, 2013a), as well as models relying on latent topic models (Vulić et al., 2011; Liu et al., 2013).

Recently, state-of-the-art BLI results were obtained by a suite of *bilingual word embedding (BWE)* models. Given source and target language vocabularies V^S and V^T , all BWE models learn a representation of each word $w \in V^S \sqcup V^T$ as a real-valued vector: $\vec{w} = [ft_1, \dots, ft_d]$, where $ft_k \in \mathbb{R}$ denotes the value for the k -th cross-lingual feature for w within a d -dimensional shared bilingual embedding space. Semantic similarity $sim(w, v)$ between two words $w, v \in V^S \sqcup V^T$ is then computed by applying a similarity function (SF), e.g. cosine (cos) on their representations in the bilingual space: $sim(w, v) = SF(\vec{w}, \vec{v}) = cos(\vec{w}, \vec{v})$.

A plethora of variant BWE models were proposed, differing mostly in the strength of bilingual supervision used in training (e.g., word, sentence, document alignments, translation pairs) (Zou et al., 2013; Mikolov et al., 2013b; Hermann and Blunsom, 2014; Chandar et al., 2014; Sjøgaard et al., 2015; Gouws et al., 2015; Coulmance et al., 2015; Vulić and Moens, 2016, inter alia). Although the BLI evaluation of the BWE models was typically performed on Indo-European languages, none of the works attempted to learn character-level representations to enhance the BLI performance.

In this work, we experiment with two BWE models that have demonstrated a strong BLI performance using only a small seed set of word translation pairs (Mikolov et al., 2013b), or document alignments (Vulić and Moens, 2016) for bilingual supervision.

It is also important to note that other word-level

translation evidence was investigated in the literature. For instance, the model of Irvine and Callison-Burch (2016) relies on raw word frequencies, temporal word variations, and word burstiness. As the main focus of this work is investigating the combination of automatically induced word-level and character-level representations, we do not exploit the whole space of possible word-level features and leave this for future work. However, we stress that our framework enables the inclusion of these additional word-level signals.

Character-Level Information for BLI For language pairs with common roots such as English-Dutch or English-Spanish, translation pairs often share orthographic character-level features, and regularities (e.g., *ideal:ideaal*, *apparition:aparición*). Orthographic translation clues are even more important in certain domains such as medicine, where words with the same roots (from Greek and Latin), and abbreviations are frequently encountered (e.g., *D-dimer:D-dimeer*, *meiosis:meiose*). When present, such orthographic clues are typically strong indicators of translation pairs (Haghighi et al., 2008). This observation was exploited in BLI, applying simple string distance metrics such as Longest Common Subsequence Ratio (Melamed, 1995; Koehn and Knight, 2002), or edit distance (Mann and Yarowsky, 2001; Haghighi et al., 2008). Irvine and Callison-Burch (2016) showed that these metrics may be used with languages with different scripts: they transliterate all words to the Latin script before calculating normalized edit distance.

BLI as a Classification Task Irvine and Callison-Burch (2016) demonstrate that BLI can be observed as a classification problem. They train a linear classifier to combine similarity scores from different signals (e.g., temporal word variation, normalized edit distance, word burstiness) using a set of training translation pairs. The approach outperforms an unsupervised combination of signals based on a mean reciprocal rank aggregation, as well as the matching canonical correlation analysis algorithm of Haghighi et al. (2008). A drawback of their classification framework is that translation signals are processed (i.e., converted to a similarity score) and weighted independently.

In contrast to their work, we propose to learn character-level representations instead of using the simple edit distance signal between candidate translations. In addition, our model identifies

translations by jointly processing and combining character-level and word-level translation signals.

3 Methodology

In this paper we frame BLI as a classification problem as it supports an elegant combination of word-level and character-level representations. Let V^S and V^T denote the sets of all unique source and target words respectively, and C^S and C^T denote the sets of all unique source and target characters. The goal is to learn a function $g : X \rightarrow Y$, where the input space X consists of all candidate translation pairs $V^S \times V^T$ and the output space Y is $\{-1, +1\}$. We define g as:

$$g(w^S, w^T) = \begin{cases} +1 & , \text{ if } f(w^S, w^T) > t \\ -1 & , \text{ otherwise} \end{cases}$$

Here, f is a function realized by a neural network that outputs a classification score between 0 and 1; t is a threshold tuned on a validation set. When the neural network is confident that w^S and w^T are translations, $f(w^S, w^T)$ will be close to 1. The reason for placing a threshold t on the output of f is twofold. First, it allows balancing between recall and precision. Second, the threshold naturally accounts for the fact that words might have multiple translations: if two target language words w_1^T and w_2^T both have high scores when paired with w^S , both may be considered translations of w^S .

Since neural network parameters are trained using a set of positive translation pairs D_{lex} , one way to interpret f is to consider it an automatically trained similarity function. For each positive training translation pair $\langle w^S, w^T \rangle$, we create $2N_s$ noise or negative training pairs. These negative samples are generated by randomly sampling N_s target language words $w_{neg,S,i}^T, i = 1, \dots, N_s$ from V^T and pairing them with the source language word w^S from the true translation pair $\langle w^S, w^T \rangle$.² Similarly, we randomly sample N_s source language words $w_{neg,T,i}^S$ and pair them with w^T to serve as negative samples. We then train the network by minimizing cross-entropy loss, expressed by Eq. (1):

²If we accidentally construct a negative pair which occurs in the set of positive pairs D_{lex} , we re-sample until we obtain exactly N_s negative samples.

$$\mathcal{L}_{ce} = \sum_{\langle w_s, w_t \rangle \in D_{lex}} \left(\log(f(w^S, w^T)) - \sum_{i=1}^{N_s} \log(f(w_{neg,T,i}^S, w^T)) - \sum_{i=1}^{N_s} \log(f(w^S, w_{neg,S,i}^T)) \right) \quad (1)$$

We further explain the architecture of the neural network and the strategy we use to identify candidate translations during prediction. Four key components may be distinguished: (1) the input layer; (2) the character-level encoder; (3) the word-level encoder; and (4) a feed-forward network that combines the output representations from the two encoders into the final classification score.

3.1 Input Layer

The goal is to exploit the knowledge encoded in both the word and character levels. Therefore, the raw input representation of a word $w \in V^S$ of character length M consists of (1) its *one-hot* encoding on the word level, labeled x_w^S ; and (2) a sequence of M one-hot encoded vectors $x_{c_0}^S, \dots, x_{c_i}^S, \dots, x_{c_M}^S$ on the character level, representing the character sequence of the word. x_w^S is thus a $|V^S|$ -dimensional word vector with all zero entries except for the dimension that corresponds to the position of the word in the vocabulary. $x_{c_i}^S$ is a $|C^S|$ -dimensional character vector with all zero entries except for the dimension that corresponds to the position of the character in the character vocabulary C^S .

3.2 Character-Level Encoder

To encode a pair of character sequences $x_{c_0}^S, \dots, x_{c_i}^S, \dots, x_{c_n}^S, x_{c_0}^T, \dots, x_{c_i}^T, \dots, x_{c_m}^T$ we use a two-layer long short-term memory (LSTM) recurrent neural network (RNN) (Hochreiter and Schmidhuber, 1997) as illustrated in Fig. 1. At position i in the sequence, we feed the concatenation of the i^{th} character of the source language and target language word from a training pair to the LSTM network. The characters are represented by their one-hot encoding. To deal with the possible difference in word length, we append special padding characters at the end of the shorter word (see Fig. 1). s_{1i} , and s_{2i} denote the states of the first and second layer of the LSTM. We found that a two-layer LSTM performed better than a shallow LSTM. The output at the final state s_{2N} is the character-level representation r_c^{ST} . We apply dropout regularization (Srivastava et al., 2014) with a keep probability of 0.5 on the output connections of the LSTM (see

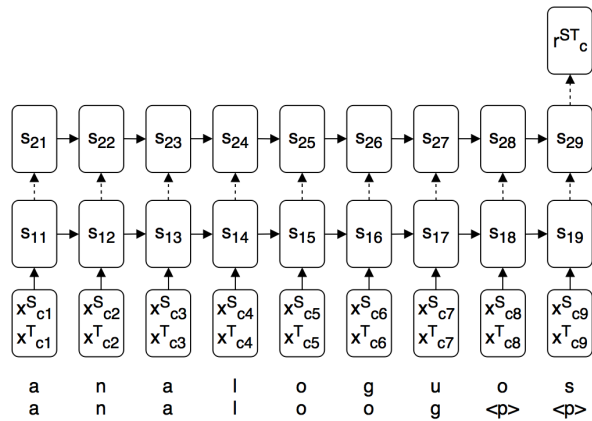


Figure 1: An illustration of the character-level LSTM encoder architecture using the example EN-NL translation pair $\langle analogous, analog \rangle$.

the dotted lines in Fig. 1). We will further refer to this architecture as CHAR-LSTM_{joint}.³

3.3 Word-Level Encoder

We define the word-level representation of a word pair $\langle w^S, w^T \rangle$ simply as the concatenation of word embeddings for w^S and w^T :

$$r_w^{ST} = W^S \cdot x_w^S \parallel W^T \cdot x_w^T \quad (2)$$

Here, r_w^{ST} is the representation of the word pair, and W^S, W^T are word embedding matrices looked up using one-hot vectors x_w^S and x_w^T . The first variant of the architecture assumes that W^S and W^T are obtained in advance using any state-of-the-art word embedding model, e.g., (Mikolov et al., 2013b; Vulić and Moens, 2016). They are then kept *fixed* when minimizing the loss from Eq. (1). In Sect. 5.3, however, we investigate another variant architecture where word embeddings are optimized *jointly* with their unsupervised context-prediction objective and the cross-entropy loss from Eq. (1).

To test the generality of our approach, we experiment with two well-known embedding models: (1) the model from Mikolov et al. (2013b), which trains monolingual embeddings using skipgram with negative sampling (SGNS) (Mikolov et al., 2013a); and (2) the model of Vulić and Moens (2016) which learns word-level bilingual embeddings from document-aligned comparable

³A possible modification to the architecture would be to swap the (unidirectional) LSTM for a bidirectional LSTM. In preliminary experiments on the development set this did not yield improvements over the proposed architecture, we thus do not discuss it further.

data (BWESG). For both models, the top layers of our proposed classification network should learn to relate the word-level features stemming from these word embeddings using a set of annotated translation pairs.

3.4 Combination: Feed-Forward Network

To combine representations on word- and character-level we use a fully connected feed-forward neural network r_h on top of the concatenation of r_w^{ST} and r_c^{ST} which is fed as the input to the network:

$$r_{h_0} = r_w^{ST} \parallel r_c^{ST} \quad (3)$$

$$r_{h_i} = \sigma(W_{h_i} \cdot r_{h_{i-1}} + b_{h_i}) \quad (4)$$

$$\text{score} = \sigma(W_o \cdot r_{h_H} + b_o) \quad (5)$$

σ denotes the sigmoid function and H denotes the number of layers between the representation layer and the output layer. In the simplest architecture, H is set to 0 and the word-pair representation r_{h_0} is directly connected to the output layer (see Fig. 2A). In this setting each dimension from the concatenated representation is weighted independently. This architecture induces undesirable patterns in the combined activation of features, and consequently does not learn generalizable relationships between source and target language inputs. On the word level, for instance, it is obvious that the classifier needs to combine the embeddings of the source and target word to make an informed decision and not merely calculate a weighted sum of them. Therefore, we opt for an architecture with hidden layers instead (see Fig. 2B). Unless stated otherwise, we use two hidden layers, while in Section 5.3 we further analyze the influence of parameter H .

3.5 Candidate Generation

To identify which word pairs are translations, one could enumerate all translation pairs and feed them to the classifier g . The time complexity of this brute-force approach is $O(|V^S| \times |V^T|)$ times the complexity of g . For large vocabularies this can be a prohibitively expensive procedure. Therefore, we have resorted to a heuristic which uses a noisy classifier: it generates $2N_c \ll |V^T|$ translation candidates for each source language word w^S as follows. It generates (1) the N_c target words closest to w^S measured by edit distance as translations, and (2) N_c target words measured closest to w^S based on the cosine distance between their word-level embeddings in a bilingual space induced by the embedding model of Vulić and Moens (2016).

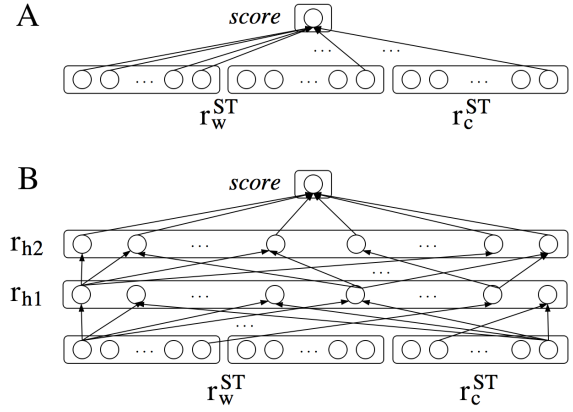


Figure 2: Illustrations of the classification component with feed-forward networks of different depths. A: $H = 0$. B: $H = 2$ (our model). All layers are fully connected.

4 Experimental Setup

Data One of the main advantages of automatic BLI systems is their portability to different languages and domains. However, current standard BLI evaluation protocols still rely on general-domain data and test sets (Mikolov et al., 2013a; Gouws et al., 2015; Lazaridou et al., 2015; Vulić and Moens, 2016, inter alia). To tackle the lack of quality domain-specific data for training and evaluation of BLI models, we have constructed a new English-Dutch (EN-NL) text corpus in the *medical* domain. The corpus contains topic-aligned documents (i.e., for a given document in the source language, we provide a link to a document in the target language that has comparable content). The domain-specific document collection was constructed from the English-Dutch aligned Wikipedia corpus available online⁴, where we retain only document pairs with at least 40% of their Wikipedia categories classified as *medical*.⁵

The simple selection heuristic ensures that the main topic of the corpus lies in the medical domain, yielding a final collection of 1198 training document pairs. Following a standard practice (Koehn and Knight, 2002; Haghghi et al., 2008; Prochasson and Fung, 2011), the corpus was then tokenized and lowercased, and words occurring less than five times were filtered out.

Translation Pairs: Training, Development, Test We construct semi-automatically a set of EN-NL translation pairs by translating all words that occur

⁴<http://linguatoools.org/tools/corpora/>

⁵https://www.dropbox.com/s/hlewabrap1b9p5n/medicine_en.txt?dl=0

in our pre-processed corpus. This process relied on Google Translate and manual corrections done by fluent EN and NL speakers. Translating the EN vocabulary yields 13,856 translation pairs in total, while the reverse process of translating the NL vocabulary yields 6,537 translation pairs. Taking the union of both lexicons results in 17,567 unique translation pairs, where 7,368 translation pairs (41.94%) have both the source and target language word occurring in our corpus.⁶

We perform a 80/20 random split of the obtained subset of 7,368 translation pairs to construct a training and test set respectively. We make another 80/20 random split of the training set into training and validation data. We note that 20.31% of the source words have more than one translation.

Word-Level Embeddings Skip-gram word embeddings with negative sampling (SGNS) (Mikolov et al., 2013b) are obtained with the `word2vec` toolkit with the subsampling threshold set to $10e-4$ and window size to 5. BWESG embeddings (Vulić and Moens, 2016) are learned by merging topic-aligned documents with length-ratio shuffling, and then by training a SGNS model over the merged documents with the subsampling threshold set to $10e-4$ and the window size set to 100. The dimensionality of all word-level embeddings is $d = 50$.

Classifier The model is implemented in Python using Tensorflow (Abadi et al., 2015). For training we use the Adam optimizer with default values (Kingma and Ba, 2015) and mini-batches of 10 examples. We used $2N_s = 10$ negative samples and we generated $2N_c = 10$ candidate translation pairs during prediction. The classification threshold t is tuned measuring F_1 scores on the validation set using a grid search in the interval $[0.1, 1]$ in steps of 0.1.

Evaluation Metric The metric we use is F_1 , the harmonic mean between recall and precision. While prior work typically proposes only one translation per source word and reports $Accuracy@1$ scores accordingly, here we also account for the fact that words can have multiple translations. We evaluate all models using two different modes: (1) *top* mode, as in prior work, identifies only one translation per source word (i.e., it is the target word with the highest classification score), (2) *all*

⁶Since we use a comparable corpus in our experiments, not all translations of the English vocabulary words occur in the Dutch part of the corpus and vice versa.

mode identifies as translation pairs all pairs for which the classification score exceeds threshold t .

5 Results and Discussion

A Roadmap to Experiments We first study automatically extracted word-level and character-level representations and their contribution to BLI in isolation (Sect. 5.1 and Sect. 5.2). It effectively means that for such single-component experiments Eq. 3 is simplified to $r_{h_o} = r_w^{ST}$ (word-level) and $r_{h_o} = r_c^{ST}$ (character-level). Following that, we investigate different ways of combining word-level and character-level representations into improved BLI models (Sect. 5.3). There, we conduct additional analyses which investigate the influence of (i) the number of hidden layers of the classifier, (ii) training data size, and (iii) other variant architectures (i.e., training word-level and character-level representations separately vs. training character-level representations jointly with the classifier vs. training all components jointly).

5.1 Experiment I: Word Level

The goal of this experiment is twofold. First, we want to analyze the potential usefulness of standard word embeddings in a classification framework. Second, we want to compare the BLI approach based on classification to standard BLI approaches that simply compute similarities in a shared bilingual space. All classification NNs are trained for 150 epochs. The results are shown in Tab. 1.

The top two rows are BLI baselines that apply cosine similarity (SIM) in a bilingual embedding space to score translation pairs. For SGNS-based embeddings, we follow (Mikolov et al., 2013b) and align two monolingual embedding spaces by learning a linear mapping using the same set of training translation pairs as used by our classification framework. The BWESG-based embeddings do not exploit available translation pairs, but rely on document alignments during training. The bottom two rows of Tab. 1 use the classification framework we proposed (CLASS).

As the main finding, we see that the classification framework using word-level features outperforms the standard similarity-based framework. BWESG in the similarity-based approach works best in *top*-mode, i.e., it is good at finding a single translation for a source word.⁷ The classification-based ap-

⁷Surprisingly, the similarity-based approach with SGNS embeddings (Mikolov et al., 2013b) reports extremely low

Representation		development		test	
		F_1 (top)	F_1 (all)	F_1 (top)	F_1 (all)
SIM	BWESG	15.71	11.56	13.43	9.84
	SGNS	0.43	0.40	0.56	0.37
CLASS	BWESG	11.51	16.02	12.12	15.09
	SGNS	17.67	20.67	17.25	19.79

Table 1: Comparison of different BLI systems which use only word-level information.

proach is consistently better in translating words with multiple translations as evident from higher *all*-mode scores in Tab. 1.

When comparing BWESG and SGNS embeddings within the classification framework, we observe that we obtain significantly better results with SGNS embeddings. A plausible explanation is that SGNS embeddings better capture properties related to the local context of a word like syntax information since they are trained with much smaller context windows.⁸ We also note that SGNS also has a practical advantage over BWESG concerning the data requirements as the former does not assume document-level alignments.

5.2 Experiment II: Character Level

Here, we compare the representation learned by the character-level encoder with manually extracted features that are commonly used. The following character-level methods are evaluated:

- CHAR-LSTM_{joint}, the output of the architecture described in Sect. 3.2
- ED_{norm}, the edit distance between the word pair normalized by the average of the number of characters of w_s and w_t as used in prior work (Irvine and Callison-Burch, 2013; Irvine and Callison-Burch, 2016).
- log(ED_{rank}), the logarithm of the rank of w_t in a list sorted by the edit distance w.r.t. w_s . This means that the target word that is nearest in edit distance w.r.t. w_s will have a feature value of $\log(1) = 0$, words that are more distant from w_s will get higher feature values.

results. A possible explanation for such results is that the model is not able to learn a decent linear mapping between two monolingual embedding spaces induced from a small monolingual corpus relying on low-frequency word translation pairs (Vulić and Korhonen, 2016). We verified the influence of low-frequency word pairs by gradually decreasing the amount of pairs in the seed lexicon, keeping only the most frequent word pairs: e.g., limiting the seed lexicon to the 1000 most frequent word pairs, we obtain better results, which are still well below other models in our comparison.

⁸Large window sizes are inherent to the BWESG model.

Representation		development		test	
		F_1 (top)	F_1 (all)	F_1 (top)	F_1 (all)
ED _{norm}		30.35	31.36	30.89	28.43
log(ED _{rank})		29.01	26.14	29.48	22.25
ED _{norm} + log(ED _{rank})		31.32	30.32	32.27	30.04
CHAR-LSTM _{joint}		33.93	35.26	33.89	34.93

Table 2: Comparison of character-level BLI methods from prior work with automatically learned character-level representations.

Target words with the same edit distance score are assigned the same rank.

- ED_{norm} + log(ED_{rank}), the concatenation of the ED_{norm} and log(ED_{rank}) features. The combined model results in a stronger baseline.

For the ED-based features we use the same classification framework. However, we use hidden layers only for ED_{norm} + log(ED_{rank}) as hidden layers do not make the the one-dimensional feature models (ED_{norm} and log(ED_{rank})) any more expressive. The ED-based models were additionally tuned by performing a grid search to find the optimal values for the number of negative samples $2N_s$ and the number of generated translation candidates $2N_c$. Both $2N_s$ and $2N_c$ are chosen from the interval [10, 100] in steps of 10 based on the performance on the validation set. The ED-based models converge quickly and were only trained for 25 epochs. For the CHAR-LSTM_{joint} representation, we use 512 memory cells per layer, we train the model for 300 epochs, and the parameters $2N_s$ or $2N_c$ were set to the default values (10) without any additional fine-tuning.

The results are displayed in Tab. 2. The overall performance is high compared to the results of the word-level models. The importance of character-level information in this data set is explained by the large amount of medical terminology and expert abbreviations (e.g., *amynoglicosides*, *aphasics*, *nystagmus*, *EPO*, *EMDR*), which due to its etymological processes, typically contain recurring morphological patterns across languages. It also further supports the need of models that are able to exploit and combine word-level and character-level features. Results also indicate that learning character-level representations from the data is beneficial as the CHAR-LSTM_{joint} model significantly outperforms the baselines used in prior work. The CHAR-LSTM_{joint} shows consistent improvements over baselines across evaluation modes, while the largest gains are again in the *all*-mode.

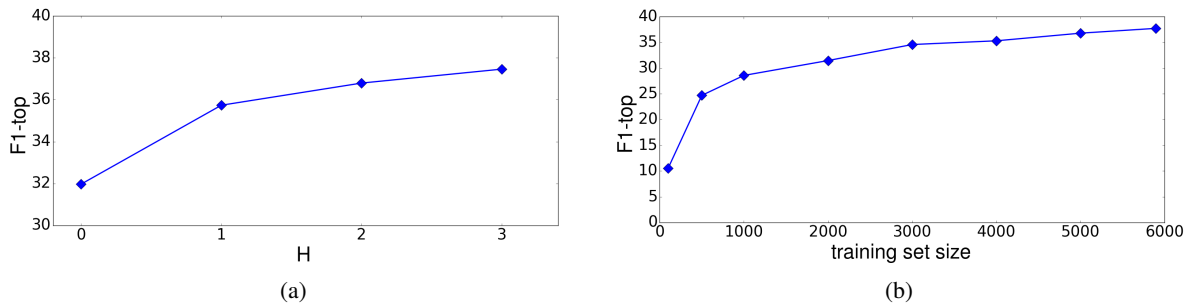


Figure 3: (a) The influence of the number of layers H between the representations and the output layer on the BLI performance; (b) The influence of the training set size (the number of training pairs).

5.3 Experiment III: Combined Model

Encouraged by the excellent results of single-component word-level and character-level BLI models in the classification framework, we also evaluate the combined model. As word-level representations we choose SGNS embeddings and the LSTM consists of 128 in memory cells in each layer in all further experiments.⁹ We compare three alternative strategies to learn the parameters of the neural network used for classification:

(1) SEPARATE: Word-level and character-level representations are trained separately. Word-level embeddings and LSTM weights for character-level representations are kept fixed when training the hidden and output layers are simply appended on top of the fixed representations.

(2) CHAR JOINT: Word-level embeddings are trained separately, while character-level representations are trained together with the hidden layers and output layer. This can encourage the network to learn new information on the character-level, different from word-level representations.

(3) ALL JOINT: Motivated by recent work (Ferreira et al., 2016) which proposed a joint formulation for learning task-specific BWEs in a document classification task, all components in our BLI framework are now trained jointly. The joint training objective now consists of two components: the context prediction objective (i.e., SGNS-style objective) and the translation objective described by Eq. (1).

The results are shown in Tab. 3. The CHAR JOINT strategy significantly improves on the best single word-level/character-level models. SEPARATE and ALL JOINT, however, do not improve on the CHAR-LSTM_{joint} model. CHAR JOINT allows the character-level representations to learn features that are complementary to word-level information,

⁹We found that in this setting, where we use both word-level and character-level representations, it is beneficial to use a smaller LSTM than in the character-level only setting.

training	development		test	
	F_1 (top)	F_1 (all)	F_1 (top)	F_1 (all)
SEPARATE	35.35	35.09	33.60	33.17
CHAR JOINT	36.78	35.85	37.73	36.61
ALL JOINT	33.02	33.75	32.86	33.31

Table 3: Results of the combined model (word-level SGNS plus CHAR-LSTM_{joint}). Three different strategies of information fusion are compared.

which seems crucial for an optimal combination of both representations. Learning word-level embeddings jointly with the rest of the network is not beneficial. This can be explained by the fact that the translation objective deteriorates the generalization capabilities of word embeddings.

Another crucial parameter is the number of hidden layers H . Fig. 3(a) shows the influence of H on F_1 in *top* mode. BLI performance increases with H . As expected, we see the largest improvement from $H = 0$ to $H = 1$. With $H = 0$ the network is not able to model dependencies between features. More hidden layers allow the network to learn more complex, abstract relations between features, resulting in an improved BLI performance.

Influence of Training Set Size In practice, for various language pairs and domains, one may have at disposal only a limited number of readily available translation pairs. Fig. 3(b) shows the influence of the size of the training set on performance: while it is obvious that more training data leads to a better BLI performance, the results suggest that a competitive BLI performance may be achieved with smaller training sets (e.g., the model reaches up to 77% of the best performance with only 1K training pairs, and > 80% with 2K pairs).

6 Conclusion and Future work

We have introduced a neural network based classification architecture for the task of bilingual lexicon

induction (BLI). We have designed, implemented and evaluated a character-level encoder in the form of a two-layer long short-term memory network and have experimented with different word-level representations. The resulting encodings were used in a deep feed-forward neural network. The results show that especially this combination of character- and word-level knowledge is very successful in the BLI task when evaluated in the medical domain.

Our novel method for learning character-level representations will raise the interest in studying character-level encoders which could be tested in different tasks where string comparisons are important. In future work, we intend to further propose and compare with alternative character-level encoding architectures, and combine additional useful BLI signals in our BLI classification framework.

Acknowledgments

This research has been carried out in the framework of the Smart Computer-Aided Translation Environment (SCATE) project (IWT-SBO 130041). IV is supported by ERC Consolidator Grant LEXICAL (no 648909). The authors are grateful to the anonymous reviewers for their helpful suggestions.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.
- Sarath A.P. Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of NIPS*, pages 1853–1861.
- Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word embeddings. In *Proceedings of EMNLP*, pages 1109–1113.
- Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of EMNLP*, pages 1285–1295.
- Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of EMNLP*, pages 1–11.
- Manaaf Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, pages 462–471.
- Daniel C. Ferreira, André F. T. Martins, and Mariana S. C. Almeida. 2016. Jointly learning to embed and predict with multiple languages. In *Proceedings of ACL*, pages 2019–2028.
- Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of ACL*, pages 414–420.
- Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of ACL*, pages 526–533.
- Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of ICML*, pages 748–756.
- Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, pages 771–779.
- Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of ACL*, pages 58–68.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of NAACL-HLT*, pages 518–523.
- Ann Irvine and Chris Callison-Burch. 2016. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics*.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.
- Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition (ULA)*, pages 9–16.
- Georgios Kontonatsios, Ioannis Korkontzelos, Jun’ichi Tsujii, and Sophia Ananiadou. 2014. Combining string and context similarity for bilingual term alignment from comparable corpora. In *Proceedings of EMNLP*, pages 1701–1712.

- Audrey Laroché and Philippe Langlais. 2010. Revisiting context-based projection methods for term-translation spotting in comparable corpora. In *Proceedings of COLING*, pages 617–625.
- Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of SIGIR*, pages 175–182.
- Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of ACL*, pages 270–280.
- Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information Processing and Management*, 41(3):523–547.
- Xiaodong Liu, Kevin Duh, and Yuji Matsumoto. 2013. Topic models+ word alignment= A flexible framework for extracting bilingual dictionary from comparable corpus. In *Proceedings of CoNLL*, pages 212–221.
- Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL*, pages 1–8.
- I. Dan Melamed. 1995. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Proceedings of Third Workshop on Very Large Corpora*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Workshop Proceedings of ICLR*.
- Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. In *CoRR*, abs/1309.4168.
- Bhaskar Mitra, Eric T. Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *CoRR*, abs/1602.01137.
- Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88.
- Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
- Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.
- Emmanuel Prochasson and Pascale Fung. 2011. Rare word translation extraction from aligned comparable documents. In *Proceedings of ACL*, pages 1327–1335.
- Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of ACL*, pages 320–322.
- Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *Proceedings of ACL*, pages 1713–1722.
- Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.
- Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of ACL*, 1:1–12.
- Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2012. Bilingual lexicon extraction from comparable corpora using label propagation. In *Proceedings of EMNLP*, pages 24–36.
- Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *Proceedings of NAACL-HLT*.
- Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*, pages 1661–1670.
- Lonneke van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of ACL*, pages 299–304.
- Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*, pages 247–257.
- Ivan Vulić and Marie-Francine Moens. 2013a. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of NAACL-HLT*, pages 106–116.
- Ivan Vulić and Marie-Francine Moens. 2013b. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of EMNLP*, pages 1613–1624.
- Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of SIGIR*, pages 363–372.
- Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.
- Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying word translations from comparable corpora using latent topic models. In *Proceedings of ACL*, pages 479–484.

- David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*, pages 200–207.
- Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag - Multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of NAACL-HLT*, pages 1307–1317.
- Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL*, pages 55–63.
- Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*, pages 1393–1398.

Grouping business news stories based on salience of named entities

Llorenç Escoter, Lidia Pivovarova, Mian Du, Anisia Katiskaya and Roman Yangarber

Department of Computer Science
University of Helsinki, Finland

first.last@cs.helsinki.fi

Abstract

In news aggregation systems focused on broad news domains, certain stories may appear in multiple articles. Depending on the relative importance of the story, the number of versions can reach dozens or hundreds within a day. The text in these versions may be nearly identical or quite different. Linking multiple versions of a story into a single group brings several important benefits to the end-user—reducing the cognitive load on the reader, as well as signaling the relative importance of the story. We present a grouping algorithm, and explore several vector-based representations of input documents: from a baseline using keywords, to a method using *salience*—a measure of importance of named entities in the text. We demonstrate that features beyond keywords yield substantial improvements, verified on a manually-annotated corpus of business news stories.

1 Introduction

We address the problem of grouping multiple versions of the same story in a system that continuously processes articles incoming from a large number of news streams. Our problem setting is PULS—an on-line information extraction (IE) system, which analyses news in the business domain.¹ PULS collects articles from over 1000 on-line sources that provide RSS feeds. Among approximately 4000–6000 articles arriving daily, some of the stories appear multiple times.

The role of the aggregation component is to cluster the articles into a set of *stories*—a story is a set of articles that report the same piece of news.

¹<http://puls.cs.helsinki.fi/>

The purpose of grouping is that when a user issues a query, the system should show one item per story, rather than one item per article, so the same fact is not shown over and over again. Information should be presented in a way that minimizes redundancy of the returned results; this implies narrowly clustering news articles that describe the same events or facts.

Another goal is to identify *trending* stories. In the business domain, if a story is globally important, it will appear in many feeds. When repeated news items are identified, the number and variety of sources covering the story is an indicator of that story's relative importance.

Thus, from the end-user's perspective, we have at least two motivations for grouping different news that describe the same story: reducing redundancy and ranking stories by importance.

The main contributions of this paper are:

- We try to identify the most effective document representation for news clustering. We demonstrate that automatically extracted *named entities* (NE) are better features than words. Moreover, considering the *salience* of NEs—a measure that combines frequency and prominence of the NE—gives further improvement in clustering performance. We introduce a novel salience weighting scheme, which in our experiments outperforms TF-IDF and raw count weighting.
- For word representation, we compare pre-trained *word2vec* vectors with vectors trained on a domain-specific news corpus. Although the corpus-specific word embeddings alone give lower performance on the clustering task, we show that they work better in combination with NE features.
- We analyze two measures for evaluation of clustering performance—Rand index and V-

measure—and a standard way of adjusting them for “chance.” We demonstrate that adjusting favors clustering with a smaller number of clusters. We propose a new type of normalization, which avoids this problem.

- We publish the *data-set*, consisting of nearly 4000 articles collected by our system for one day, grouped into clusters manually. The data-set represents a real task, which the aggregation component must solve daily. We also provide a command-line *annotation tool*, to facilitate manual clustering. We also publish the *word embeddings*.

Although in this paper we deal only with business news, we consider redundancy and the tendency to repeat the more important events to be general properties of news streams. Thus, we believe that our task and the proposed solution generalize to many other news-monitoring settings.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents the NE extraction system and introduces salience for NE weighting. Section 4 describes the algorithm and features. Section 5 describes the data and annotation process. Section 6 discusses evaluation methods and results. Section 7 contains conclusions and future work.

2 Related work

A general overview of text clustering techniques can be found in (Aggarwal and Zhai, 2012). Many results on document clustering are published in IR literature, where these techniques are used to cluster search results (Carpineto et al., 2009).

News are clustered for various purposes: finding breaking news in streams (Kumaran and Allan, 2004), linking duplicates or articles about the same story (Vadrevu et al., 2011), tracking threads of news over time (van Erp et al., 2014; Azzopardi and Staff, 2012; Steinberger and Pouliquen, 2008), or facilitating access to information (Zhang et al., 2013; Toda and Kataoka, 2005).

The main techniques for clustering documents are: agglomerative clustering (Steinberger and Pouliquen, 2008) and partitioning clustering, such as k-means, buckshot, and *fractionation* (Azzopardi and Staff, 2012; Sankaranarayanan et al., 2009; Cutting et al., 1992). Hierarchical agglomerative clustering is commonly used in practice, though in general it has a complexity of $O(n^2 \log(n))$, (Berkhin, 2006). All objects start

in their own, trivial cluster. The closest pair of clusters is merged, iteratively, until the hierarchy is complete. Partitional algorithms can also be used to create a hierarchical solution, e.g., *bisecting k-means*, which is better than standard k-means and comparable to agglomerative hierarchical approaches, (Steinbach et al., 2000).

On the other hand, determining the number of clusters might be a tricky task for partitional algorithms (Gialampoukidis et al., 2016).

Suffix Tree Clustering (STC) is a linear-time algorithm based on identifying common phrases within groups of documents, (Zamir and Etzioni, 1999). Spectral clustering models the documents as an undirected graph, where each node represents a document, assigns a similarity between documents as a weight on the edges, and tries to find the best cuts of the graph, (Shi and Malik, 2000). Xu et al. (2003) identify document clusters in the latent semantic space derived by non-negative factorization of the term-document matrix of the given corpus.

A common procedure for agglomerative clustering (also used in this paper) can be summarized as follows: convert documents into a vector representation, then use a metric to compute pairwise similarity between documents—often, cosine similarity. In this procedure, clustering quality crucially depends on document representation.

Traditionally, a common way of representing documents for clustering is by a vector of TF-IDF weights for each keyword, e.g., (Iglesias et al., 2016; Azzopardi and Staff, 2012; Vadrevu et al., 2011). Steinberger and Pouliquen (2008) use log-likelihood (LL) for weighting keywords rather than TF-IDF. LL statistics can be computed for each word in the corpus, relative to a separate, “reference” corpus (Rayson and Garside, 2000). Staff et al. (2015) claim that for search results, using raw term frequencies outperforms TF-IDF. Recent lines of research use word embeddings (Mikolov et al., 2013b) to represent documents. Sophisticated deep learning algorithms can also be applied to text clustering (Xu et al., 2015), but to date they require labeled training data, while the method proposed in this paper is unsupervised.

In contrast to bag-of-words (BOW) schemes, named entities (NEs) can be used as features (Montalvo et al., 2012). In most cases, NEs are also weighted according to TF-IDF (Toda and Kataoka, 2005) or its variants (Cheng et al., 2012;

Kiritoshi and Qiang, 2016).

Kumaran and Allan (2004) combined three vector representations for a document, namely: all words, NEs, and all words except NEs. This is similar to the series of experiments in this paper; the difference is that Kumaran and Allan (2004) used TF-IDF in all three cases, while we compare TF-IDF with several alternatives.

Popular clustering data sets target much coarser categorization tasks. For instance, 20 News-group² and Reuters' RCV1³ categorize news into business sectors such as "Fruit Growing" or "comp.graphics." TDT2⁴ classifies news related to certain topics—a phenomenon or a big event—such as "Asian Economic Crisis" or "1998 Winter Olympics." Our task focuses on more focused groups; stories about business activities that occurred within a given industry sector or that are related to a broader phenomenon are not considered the same story. If the same entities engage in two different activities, we consider that as two distinct stories. Therefore, we manually annotated a sample of our corpus, which is more suitable for evaluating our methods.

3 Named Entities and Saliency

We use a Named Entity Recognition module as part of the PULS news monitoring system. (Yangarber and Steinberger, 2009; Huttunen et al., 2013; Du et al., 2016) It uses patterns and rules to extract NEs; currently the system uses about a thousand patterns, some of which were learned (Yangarber, 2003) and some manually constructed. The system assigns a *type* to each NE—company, person, product, etc.—but the NE types are not used for grouping to reduce the effect of mistakes in analysis, e.g., when an entity is classified with different types across multiple documents. Rather, we consider clustering to be an earlier step in the overall processing pipeline (Yangarber, 2006). Ji and Grishman (2008) show that performance of an IE system can be improved by using clusters of topically-related documents. In PULS we use grouping to improve NE classification: we assign each entity a type based on the majority within the set of clustered documents.

²<http://www-2.cs.cmu.edu/~TextLearning/datasets.html>

³<http://about.reuters.com/researchandstandards/corpus/>

⁴<https://catalog.ldc.upenn.edu/LDC2001T57>

Our definition of *saliency* relies on the general nature of news articles. Authors typically mention the main event in the title, in condensed form; then, the main information is elaborated in the first few sentences, followed by further detail and background. Thus, the most important NEs are mentioned early in the text and then repeated, whereas less important NEs are mentioned in the later paragraphs and are less frequent.

We compute *saliency* as a combination of *prominence* and *frequency* of an entity in a document. Prominence captures the importance of the *first* mention of entity e in document d :

$$prominence(e, d) = \frac{ns(d) - fs(e, d)}{ns(d)}$$

where $ns(d)$ is the total number of sentences in the document, $fs(e, d)$ is the number of the sentence (starting at zero) of the first mention of e in d . Thus, the prominence of entities mentioned in the title is 1. Prominence also takes into account the total length of the document, to capture diversity of news sources in the collection. For example, the second sentence in a two-page article is more important than the second sentence in a short article, where all crucial information must be condensed at the very beginning.

Frequency is the ratio of mentions of a given NE over all NE mentions:

$$frequency(e, d) = \frac{C(e, d)}{\sum_{e' \in NE(d)} C(e', d)}$$

where $C(e, d)$ is the count of e in d , $NE(d)$ is the set of all NEs in d . Note that we compute the NE frequency relative to the other NEs only and ignore all the other words in the document, since NEs and common words have rather different distributions: important terms are usually repeated more times than names.

We define saliency as the geometric mean of prominence and frequency:

$$S(e, d) = \sqrt{prominence(e, d) \cdot frequency(e, d)}$$

Saliency lies between 0 and 1, but the saliences in a document need not add up to one—there may be more than one salient entity in the document, or none. In the business domain, the majority of events involve some NEs (often, companies).

We make extensive use of saliency in the PULS system to aggregate and present information to

users. For example, we represent each group as a list of salient companies, with the least salient companies removed. Similarly, when a user searches for a name, the system returns only documents where the entity salience is above a certain threshold.

We compare salience to two other weighting strategies: namely, frequency alone, and TF-IDF.

4 Clustering algorithm

The experiments follow the same structure. We start with a collection of documents and transform it into a collection of vectors, by one of the methods described below. We apply agglomerative clustering to the collection of document vectors, using cosine distance between vectors. The agglomerative algorithm produces a dendrogram with the documents as leaves, and we obtain a clustering by cutting at a distance threshold $\theta \in (0, 1)$. We use the *complete* metric, meaning that the distance between clusters A and B is the *maximum* of the distances between any two vectors in A and B (Aggarwal and Zhai, 2012). The threshold θ imposes a limit on the maximum distance between any two documents in a cluster.

4.1 Mapping documents to \mathbb{R}^k

To represent documents as vectors we use two types of features: all words, or named entities. For words, we use three representations: TF-IDF, *word2vec* embeddings pre-trained on a large general corpus, and embeddings trained on our business-news corpus. For NEs, we use raw counts, TF-IDF and salience. Thus, we experiment with six vector representations.

Word-based representations: For each word w , we compute TF-IDF as:

$$\text{TF-IDF}(w, d) = \frac{C(w, d)}{\sum_{w' \in W(d)} C(w', d)} \cdot \log \frac{|D|}{|D_w|}$$

where $C(w, d)$ counts how many times the word w appears in document d , $W(d)$ is all words in d , D is all documents in the corpus, and $|D_w|$ are all documents in D which contain word w . Then the vector representation for the document is:

$$\text{TF-IDF}(d) = \sum_{w \in W(d)} \text{TF-IDF}(w, d) \hat{u}_w$$

where \hat{u}_w is the *one-hot* vector, whose length is the size of the vocabulary, and which contains zeros in all positions but the one corresponding to

w_1	w_2	Google News	Business
jump	climb	0.55	0.85
recall	remember	0.43	0.13

Table 1: Cosine similarity for sample word vectors.

w . The vector $\text{TF-IDF}(d)$ contains TF-IDF values for its words and zeros in all other positions. We use only content words—nouns, adjectives and verbs—in the TF-IDF representations.

Another approach is to represent each word as a vector in a low-dimensional vector space. We can then represent documents by adding their corresponding word vectors. We use word vectors produced by the CBOW approach—continuous bag-of-words (Mikolov et al., 2013a). The vector representing document d is then:

$$\text{CBOW}(d) = \sum_{w \in W(d)} C(w, d) e_w$$

where e_w is the *embedding* vector representing w .

In this paper we use the “standard” word2vec embeddings built on the Google News data-set⁵ (referred to as “CBOW-st”), and embeddings trained on our business-news corpus “CBOW-b”. Our corpus is relatively small (4.5 million documents), but it contains only documents relevant to business news. We do not know *a priori* which set of embeddings is more suitable for our task. Although the two embeddings produce similar results, the resulting word vectors have noticeable differences, as can be seen in Table 1. The business-domain embeddings for *jump* and *climb* are much closer than in the general corpus, since both are used to denote *increases*; meanwhile, embeddings for *recall* and *remember* are much closer in the general corpus, because, in the business domain, *recall* frequently refers to product recalls.

Named entity-based representations: Another natural representation for a document d can be obtained by using only named entity counts:

$$\text{NEC}(d) = \sum_{e \in \text{NE}(d)} C(e, d) \hat{u}_e$$

where $\text{NE}(d)$ is the set of all named entities in document d ,⁶ and \hat{u}_e is the one-hot vector. TF-IDF for NEs is computed in the same way as for keywords

⁵code.google.com/archive/p/word2vec

⁶Here we use counts instead of frequencies since there are equivalent when cosine similarity is used.

but using only NEs, ignoring common words. We refer to this representation as **NE-TFIDF**. Finally, a salience-based document representation leverages the *salience* S of the NEs, described in Section 3:

$$\text{NES}(d) = \sum_{e \in \text{NE}(d)} S(e, d) \hat{u}_e$$

4.2 Combining representations

Named entities are crucial for news clustering. In some reporting, journalists use standardized language and even article templates to describe similar events. In such cases, the article’s NEs are the only way to obtain a meaningful document similarity. On the other hand, NEs can be misleading, e.g., if two different events take place in the same location (see (Kumaran and Allan, 2004) for examples in both kinds of problems). Thus, we try to *combine* NE and other textual features.

To that end, we use the best-performing methods in their respective categories—see results in Section 6—namely CBOW for words and salience for NEs. We use two methods of combining of CBOW and NES representation: *juxtaposition* and the *AND* function.

Juxtaposing means simply appending together the vectors corresponding to the NEs and the keywords, to form longer document vectors:

$$\text{NES_CBOW}_\alpha(d) = \left[\alpha k_d \frac{\text{NES}(d)}{\|\text{NES}(d)\|}, \frac{\text{CBOW}(d)}{\|\text{CBOW}(d)\|} \right] \quad (1)$$

where first, we normalize both representations by their respective Euclidean distances, $\|\cdot\|$; then, we scale by α , which controls the relative weight of NES vs. CBOW. We further scale NES by k_d —the number of NEs found in d . The rationale behind this is that if a document contains more NEs, then the NES representation conveys more information; whereas if d has only one NE, then grouping should rely much more on the keywords. We apply the same agglomerative clustering procedures as in other experiments to these juxtaposed vectors. We experiment with both vector representations, CBOW-st and CBOW-b.

The second method, similar to that used in (Kumaran and Allan, 2004), requires that both word distance *and* NE distance should be sufficiently close—closer than corresponding thresholds. In this case, we cannot use the *complete* linkage metric since a maximum of distances is not defined if the distance is a *pair* of numbers. Thus, unlike all

other experiments described in this paper, for the *AND* combination method we use the *single* metrics (Aggarwal and Zhai, 2012). This is equivalent to finding connected components of the graph where the nodes are documents and there is an edge between two nodes if and only if both distances are below their respective thresholds.

5 Data and annotation scheme

From our business news corpus (Pivovarov et al., 2013) we selected one “typical” day for annotation, with a total of 3959 documents.⁷ We manually annotated all of these documents via a specialized interface, which displays documents pairwise and allows an annotator to make three main decisions: documents can be

- Grouped: if their main stories are the same.
- Not grouped: if their stories are not the same.
- Partially grouped: if their main stories are not the same, but may partially overlap. For example, one article might mention the other’s main story toward the end.

The interface provides other helpful options, for example, the annotator can use regular expressions to search for all documents similar to a given one; another option is to mark the document as invalid if the document is *malformed*.⁸

We use a triangular matrix M to keep track of the pairwise relations among documents. Since annotation is an extremely time-consuming process, the key aim is to reduce the amount of data shown to the annotator as far as possible. We initialize M by pre-marking all pairs of documents that do have no NE in common as un-grouped; this decision may later be reversed by the annotator.

Another idea that allows us to minimize manual work is *decision propagation*. Document grouping—that is, documents having the same main story—can be viewed as an equivalence relation. This means that (ideally) only one member of a group needs to be checked against a member of another group to decide whether both groups

⁷Some sites publish “summary” articles, which contain an overview of 10 or more (possibly unrelated) stories, with as little as one sentence per story. In this paper, summaries are filtered out, to make the grouping task well defined. Filtering is performed by a simple segmentation algorithm, which checks whether the text is separable into contiguous segments, containing *non-overlapping* sets of named entities.

⁸This includes documents where some *broken* content was retrieved, such as a login page or advertisement.

should be merged. Every time a user groups two documents, the decision is propagated so that two corresponding groups are merged, and other pairs from the merged group are never shown to the user. Sometimes this leads to a contradiction, including cases when the initialization was wrong—i.e., when initialization suggests that two documents cannot be in the same group, but the annotator decides to merge two groups they belong to. In such a case, the annotator is warned and has to resolve the contradiction. Negative decisions can also be propagated: when an annotator marks a pair of documents as ungrouped, this affects all documents in both groups. Partial relations, on the other hand, are not equivalence relations, and require more manual annotation.

Our annotation tool keeps track of annotators’ decisions, U , and reconstructs the annotation from them. This process can be viewed as applying each $u \in U$ to successive versions of M :

$$M_i = u_i(M_{i-1})$$

In this scenario, mistakes—which will appear as contradictions in M —are much easier to detect and correct. Given U that generates contradictions, a minimal subset $U' \subset U$ that generates the same contradiction can be more easily inspected. Then, the offending input can be corrected and we can proceed with the annotation.⁹

In total four members of our team were involved in the annotation process. Most of the instances were annotated by one person. In the beginning of the process we annotated several cases together and discussed difficult ones to work out general guidelines. Annotators also checked some random part of others’ annotations, and corrected several cases during error analysis, by looking at misclassified instances with the highest confidence.¹⁰

Of the 3959 documents annotated in this manner, all documents marked as either invalid or *partially related* to others were removed (402 documents), leaving 3557 documents that can be grouped unambiguously. This constitutes the ground-truth clustering against which we test our system. Figure 1 shows how the documents are distributed among cluster sizes: the vast majority of them (2249) are in a cluster by themselves,¹¹

⁹We arrived at this annotation scheme through trial and error, since annotating thousands of documents is a complex and tedious task. This seems to be an effective approach.

¹⁰Overall, the annotation process spanned across two calendar months.

¹¹Leftmost bar is cut off at 500 to improve readability.

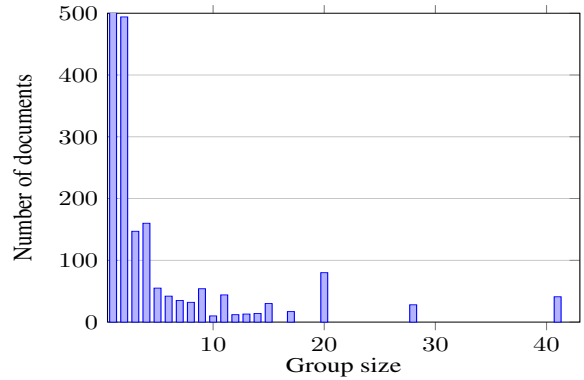


Figure 1: Distribution of annotated data (the leftmost bar goes up to 2249: 2249 documents are not clustered with any other document)

and the rest form larger ones. This is expected, as most of the data has been gathered from specialized business RSS feeds (mining news, dairy news, and so on); these sources usually report all news related to their industry including less important events that do not appear in other sources.

6 Evaluation and results

6.1 Evaluation methods

We evaluate our resulting clusters using Rand index (RI) and V-Measure. Rand index considers all possible pairs of documents, and is the proportion of correctly classified pairs—grouped or ungrouped—among all pairs (Rand, 1971). RI can be adjusted for chance, as described in (Hubert and Arabie, 1985):

$$ARI(S_{rep}) = \frac{RI(S_{rep}) - \mathbb{E}[RI(S_{chance})]}{1 - \mathbb{E}[RI(S_{chance})]} \quad (2)$$

where S_{rep} is the clustering strategy based on a document representation rep — rep is one of the representation strategies described in Section 4. The strategy S_{chance} is a *random* clustering strategy; $RI(S)$ is the RI of applying strategy S to the data; and \mathbb{E} is expectation, which is estimated as described in (Hubert and Arabie, 1985).

V-Measure is the harmonic mean of homogeneity (H) and completeness (C) (Rosenberg and Hirschberg, 2007).

$$H = 1 - \frac{\mathbb{H}(C|K)}{\mathbb{H}(C)} \quad C = 1 - \frac{\mathbb{H}(K|C)}{\mathbb{H}(K)} \quad V = \frac{2HC}{H + C}$$

where \mathbb{H} denotes entropy, K are predicted labels,

and C are the true labels. Completeness, homogeneity and V-measure are analogous to recall, precision and F-measure respectively.

Figures 2 and 3 show the measures we wish to optimize, namely the Rand index (*unadjusted for chance*) and the V-Measure. RI adjusted for chance (ARI) is shown in Figure 4.

From figures 2 and 3 it is relatively difficult to see the maxima of the measures. In both, most of the *interesting* information lies in a very small region, which is difficult to visualize. If we zoomed into these regions sufficiently, we would find that the maxima in these figures correspond to similar values θ , which are different from the maxima in Figure 4. This highlights several problems.

Adjustment for chance depends on the number of clusters, which in turn depends on the threshold value (θ); a random clustering that produces a very large number of clusters will have a very high RI value (which is the adjustment penalty) and the ARI will be very low, because the penalty for the adjustment is high. Therefore, the ARI measure favors higher values of θ , where the number of resulting clusters will be smaller.

We consider this a shortcoming of the ARI measure and propose another function to maximize. We rather adjust for the **naïve** strategy, $S_{\text{naïve}}$, which assigns each document to its own cluster. In other words, we try to measure what is the gain of clustering some documents compared to “doing nothing.”

We transform Equation 2 to adjust for $S_{\text{naïve}}$ rather than chance. Suppose $f(S)$ is a scoring measure for a clustering strategy S ; in our case, f is RI or V-measure. Now, $f(S) \in [0, 1]$, and 1 is the perfect score, and 0 is the worst score. We can adjust f as follows:

$$\begin{aligned} \hat{f}_{\text{naïve}}(S_{\text{rep}}) &= \frac{f(S_{\text{rep}}) - \mathbb{E}[f(S_{\text{naïve}})]}{1 - \mathbb{E}[f(S_{\text{naïve}})]} \\ &= \frac{f(S_{\text{rep}}) - f(S_{\text{naïve}})}{1 - f(S_{\text{naïve}})} \end{aligned} \quad (3)$$

Equation 3 adjusts the score for the naïve strategy—which shows how much better than the naïve the given strategy performs; if it performs worse than naïve adjusted score is less than zero.

The naïve strategy produces high scores: V-measure of 0.965 and RI of 0.9993. Homogeneity for the naïve strategy is 1, and completeness for our corpus is also quite high because the majority of documents do not belong to any group, as

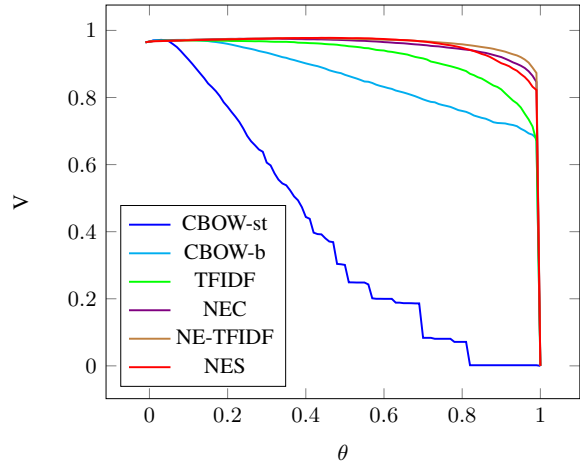


Figure 2: V-Measure

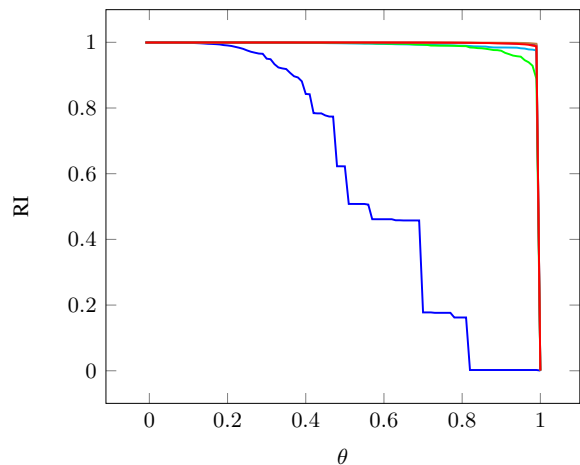


Figure 3: Rand index

shown in Figure 1. Rand index considers all possible pairings and yields a high score since most must belong to different clusters.

Using the naïve adjustment, Figures 5 and 6 show a much clearer picture of how each document representation behaves.¹² The figures show that the two measures—RI and V-measure—behave similarly, and reach their maxima at very similar values of θ . Because the measures indicate the same maximum, we do not need to prefer one measure over the other.

6.2 Results

As seen in Figure 5 and 6, the NE-based strategies outperform the word-based ones. Even the worst-performing NE-based measure (raw count, NEC) is better than the word-based strategies. TF-IDF, which is the most frequently mentioned strategy in the literature, outperforms raw counts. We can

¹²In these figures we show only positive values.

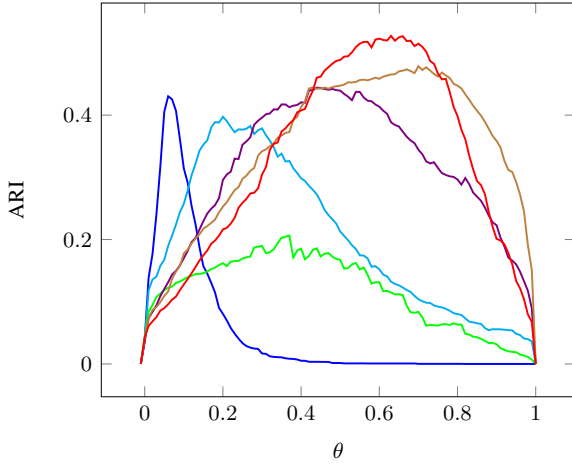


Figure 4: Adjusted Rand index

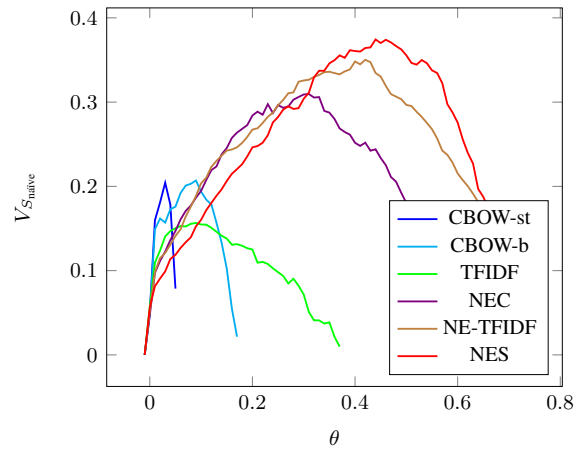


Figure 6: V Measure adjusted for S_{naive}

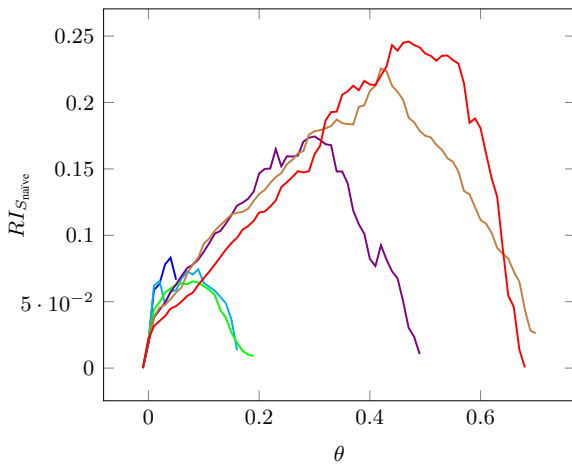


Figure 5: Rand index adjusted for S_{naive}

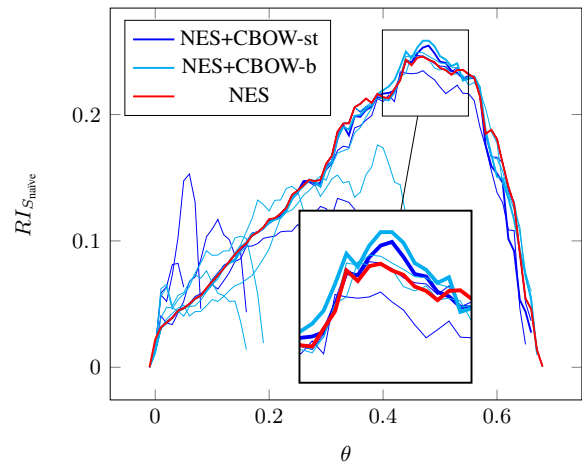


Figure 7: Rand index adjusted for S_{naive}

argue for and against TF-IDF. For example, it is clear that locations that rarely pop up in news are more informative than popular country names. On the other hand, big companies, such as Google, are involved in many different activities and often appear in the news, which should not affect their relevance in a particular event. The best-performing measure, which is based on *saliency* (NES), completely ignores the overall distribution of NEs in the corpus. However, it takes into account the *position* of the entity mentions in the text, and manages to outperform both raw counts and TF-IDF.

Among word-based measures, embeddings—CBOw-st and CBOw-b—outperform TF-IDF, and pre-trained embeddings, CBOw-st, are slightly better than the ones trained on our small business corpus, CBOw-b. It is also interesting how concave the CBOw plots are, as can be seen in Figures 5 and 6; this shows that the embedding representation has a clear, well-defined, threshold.

Figures 7 and 8 show the measures obtained by combining embedding-based representations and saliency, according to Equation 1—juxtaposing combination method. We tested several values of α , on a logarithmic scale from $\frac{1}{1000}$ to 1000, some of them are shown using thin blue lines, dark for CBOw-st and light for CBOw-b. The thick lines present the best performing combinations, which correspond to $\alpha = 1$ for CBOw-st and $\alpha = 0.5$ for CBOw-b; the red curves present the values obtained by NES in figures 5 and 6.

It can be seen, from Figures 7 and 8, that, when combined with saliency, the embeddings trained on our small domain-specific corpus outperform those trained on a much larger general corpus, even though CBOw-b alone performs worse than CBOw-st. It is interesting that, in the case of business-specific embeddings, it is better to give less weight to the NE features: the best α for CBOw-st is half of the best α for CBOw-b.

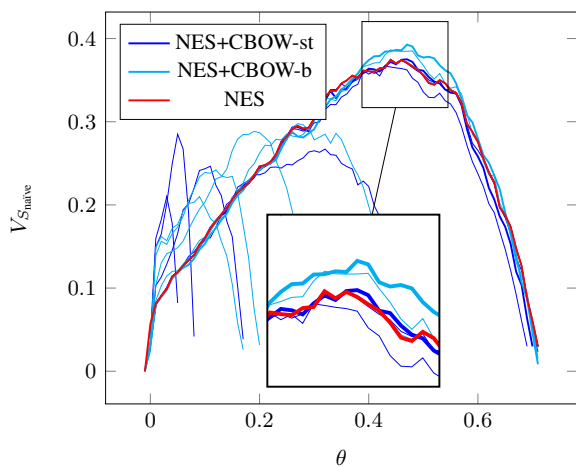


Figure 8: V-Measure adjusted for S_{naive}

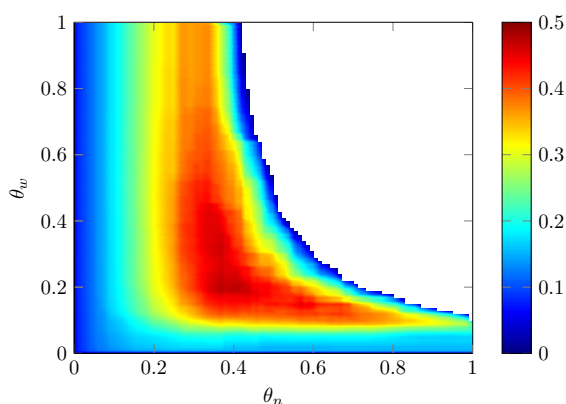


Figure 9: V-Measure adjusted for S_{naive}

However, as can be seen from these figures, juxtaposed representations does not yield significant improvements over simply using a single set of features. Figures 9 and 10 show, respectively, the V-measure and Rand-index for CBOW-b and NES combined using the *AND* function: two documents are in the same group if and only if both distances are below their corresponding thresholds— θ_n for NEs and θ_w for words. Since we are optimizing these two parameters, we plot the results in a heat map. It can be seen that this approach to combination yields a significant improvement: up to 0.39 for S_{naive} -adjusted Rand index and 0.47 for V-measure.

This significant improvement can be explained by the distribution of our corpus, presented in Figure 1, and by the fact that we use two representations of documents with different information. By using the *AND* function to combine them, we can filter out the cases where using only one representation would result in a false positive. In other

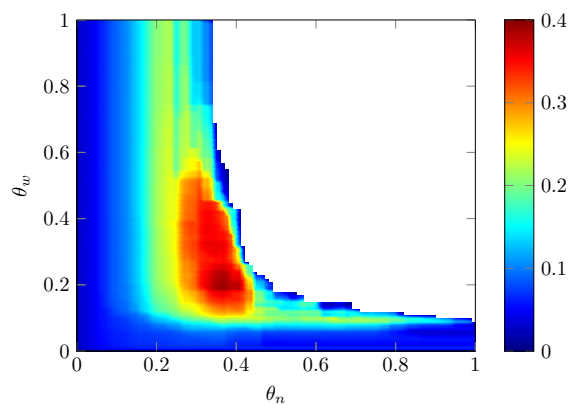


Figure 10: Rand index adjusted for S_{naive}

words, two documents are grouped only if both their names and their common keywords are similar. We hypothesize that this can be a reasonable method for event-based clustering of news streams: the trending events are reported in many sources, while each source tries to produce some unique content. On the other hand, this may be not an appropriate strategy for topic classification.

7 Conclusions and future work

We have shown how considering the relative importance of named entities, in the form of *saliency*, can be used to improve detection of related stories in different news articles. We have introduced an effective adjustment for the clustering metrics, and a method for combining different document vector representations, which outperforms the base representations alone. We make public the annotation interface, the news data, and the word embeddings used in this work, on our project page.¹³

We plan to explore other, more *user-oriented* metrics, which could take into account what a potential user might expect from a news-aggregating system. Other representations using the relative importance of named entities in a given news article should also be considered, such as a continuous-vector representation for documents where named entities play a role. Zhao and Karypis (2002) claim that agglomerative clustering may not be the best algorithm for this kind of task; therefore we plan to explore how the representations behave under other clustering algorithms.

¹³<http://puls.cs.helsinki.fi/grouping>

References

- Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer.
- Joel Azzopardi and Christopher Staff. 2012. Incremental clustering of news reports. *Algorithms*, 5(3):364–378.
- Pavel Berkhin. 2006. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.
- Claudio Carpineto, Stanislaw Osinski, Giovanni Romano, and Dawid Weiss. 2009. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17.
- Jia Cheng, Jingyu Zhou, and Shuang Qiu. 2012. Fine-grained topic detection in news search results. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 912–917. ACM.
- Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM.
- Mian Du, Lidia Pivovarova, and Roman Yangarber. 2016. PULS: natural language processing for business intelligence. In *Proceedings of the 2016 Workshop on Human Language Technology*, pages 1–8. Go to Print Publisher.
- Ilias Gialampoukidis, Stefanos Vrochidis, and Ioannis Kompatsiaris. 2016. A hybrid framework for news clustering based on the dbSCAN-martingale and lda. In *Machine Learning and Data Mining in Pattern Recognition*, pages 170–184. Springer.
- Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.
- Silja Huttunen, Arto Vihavainen, Mian Du, and Roman Yangarber. 2013. Predicting relevance of event extraction for the end user. In Thierry Poibeau, Horacio Saggion, Jakub Piskorski, and Roman Yangarber, editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 163–176. Springer Berlin.
- José Antonio Iglesias, Alexandra Tiemblo, Agapito Ledezma, and Araceli Sanchis. 2016. Web news mining in an evolving framework. *Information Fusion*, 28:90–98.
- Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*, pages 254–262.
- Keisuke Kiritoshi and M. A. Qiang. 2016. Named entity oriented difference analysis of news articles and its application. *IEICE TRANSACTIONS on Information and Systems*, 99(4):906–917.
- Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM.
- Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *NIPS*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Soto Montalvo, Víctor Fresno, and Raquel Martínez. 2012. Nesm: a named entity based proximity measure for multilingual news clustering. *Procesamiento del lenguaje natural*, 48:81–88.
- Lidia Pivovarova, Silja Huttunen, and Roman Yangarber. 2013. Event representation across genre. In *Proceedings of the 1st Workshop on Events: Definition, Detection, Coreference, and Representation*, NAACL HLT.
- William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.
- Paul Rayson and Roger Garside. 2000. Comparing corpora using frequency profiling. In *Proceedings of the workshop on Comparing Corpora*, pages 1–6. Association for Computational Linguistics.
- Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420.
- Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. 2009. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 42–51. ACM.
- Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.
- Chris Staff, Joel Azzopardi, Colin Layfield, and Daniel Mercieca. 2015. Search results clustering without external resources. In *2015 26th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 276–280. IEEE.

- Michael Steinbach, George Karypis, Vipin Kumar, et al. 2000. A comparison of document clustering techniques. In *KDD workshop on text mining*, pages 525–526. Boston.
- Ralf Steinberger and Bruno Pouliquen. 2008. NewsExplorer—combining various text analysis tools to allow multilingual news linking and exploration. *Lecture notes for the lecture held at the SORIA Summer School Cursos de Tecnologias Linguísticas*.
- Hiroyuki Toda and Ryoji Kataoka. 2005. A search result clustering method using informatively named entities. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 81–86. ACM.
- Srinivas Vadrevu, Choon Hui Teo, Suju Rajan, Kunal Punera, Byron Dom, Alexander J Smola, Yi Chang, and Zhaohui Zheng. 2011. Scalable clustering of news search results. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 675–684. ACM.
- Marieke van Erp, Gleb Satyukov, Piek Vossen, and Marit Nijssen. 2014. Discovering and visualising stories in news. In *LREC*, pages 3277–3282.
- Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267–273. ACM.
- Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 62–69.
- Roman Yangarber and Ralf Steinberger. 2009. Automatic epidemiological surveillance from on-line news in MedISys and PULS. In *Proceedings of IMED-2009: International Meeting on Emerging Diseases and Surveillance*, Vienna, Austria.
- Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.
- Roman Yangarber. 2006. Verification of facts across document boundaries. In *Proceedings of the International Workshop on Intelligent Information Access (IIIA-2006)*, Helsinki, Finland.
- Oren Zamir and Oren Etzioni. 1999. Grouper: a dynamic clustering interface to web search results. *Computer Networks*, 31(11):1361–1374.
- Jiwei Zhang, Qiuyue Dang, Yueming Lu, and Songlin Sun. 2013. Suffix tree clustering with named entity recognition. In *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, pages 549–556. IEEE.
- Ying Zhao and George Karypis. 2002. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM.

Very Deep Convolutional Networks for Text Classification

Alexis Conneau
Facebook AI Research
aconneau@fb.com

Holger Schwenk
Facebook AI Research
schwenk@fb.com

Yann Le Cun
Facebook AI Research
yann@fb.com

Loïc Barrault
LIUM, University of Le Mans, France
loic.barrault@univ-lemans.fr

Abstract

The dominant approach for many NLP tasks are recurrent neural networks, in particular LSTMs, and convolutional neural networks. However, these architectures are rather shallow in comparison to the deep convolutional networks which have pushed the state-of-the-art in computer vision. We present a new architecture (VD-CNN) for text processing which operates directly at the character level and uses only small convolutions and pooling operations. We are able to show that the performance of this model increases with the depth: using up to 29 convolutional layers, we report improvements over the state-of-the-art on several public text classification tasks. To the best of our knowledge, this is the first time that very deep convolutional nets have been applied to text processing.

1 Introduction

The goal of natural language processing (NLP) is to process text with computers in order to analyze it, to extract information and eventually to represent the same information differently. We may want to associate categories to parts of the text (e.g. POS tagging or sentiment analysis), structure text differently (e.g. parsing), or convert it to some other form which preserves all or part of the content (e.g. machine translation, summarization). The level of granularity of this processing can range from individual characters to subword units (Sennrich et al., 2016) or words up to whole sentences or even paragraphs.

After a couple of pioneer works (Bengio et al. (2001), Collobert and Weston (2008), Collobert et al. (2011) among others), the use of neural networks for NLP applications is attracting huge in-

terest in the research community and they are systematically applied to all NLP tasks. However, while the use of (deep) neural networks in NLP has shown very good results for many tasks, it seems that they have not yet reached the level to outperform the state-of-the-art by a large margin, as it was observed in computer vision and speech recognition.

Convolutional neural networks, in short *ConvNets*, are very successful in computer vision. In early approaches to computer vision, handcrafted features were used, for instance “*scale-invariant feature transform (SIFT)*” (Lowe, 2004), followed by some classifier. The fundamental idea of ConvNets (LeCun et al., 1998) is to consider feature extraction and classification as one jointly trained task. This idea has been improved over the years, in particular by using many layers of convolutions and pooling to sequentially extract a *hierarchical representation* (Zeiler and Fergus, 2014) of the input. The best networks are using more than 150 layers as in (He et al., 2016a; He et al., 2016b).

Many NLP approaches consider words as basic units. An important step was the introduction of continuous representations of words (Bengio et al., 2003). These *word embeddings* are now the state-of-the-art in NLP. However, it is less clear how we should best represent a sequence of words, e.g. a whole sentence, which has complicated syntactic and semantic relations. In general, in the same sentence, we may be faced with local and long-range dependencies. Currently, the mainstream approach is to consider a sentence as a sequence of tokens (characters or words) and to process them with a recurrent neural network (RNN). Tokens are usually processed in sequential order, from left to right, and the RNN is expected to “*memorize*” the whole sequence in its internal states. The most popular and successful RNN variant are certainly LSTMs (Hochreiter and Schmid-

Dataset	Label	Sample
Yelp P.	+1	Been going to Dr. Goldberg for over 10 years. I think I was one of his 1st patients when he started at MHMG. Hes been great over the years and is really all about the big picture. [...]
Amz P.	3(5)	I love this show, however, there are 14 episodes in the first season and this DVD only shows the first eight. [...]. I hope the BBC will release another DVD that contains all the episodes, but for now this one is still somewhat enjoyable.
Sogou	"Sports"	ju4 xi1n hua2 she4 5 yue4 3 ri4 , be3i ji1ng 2008 a4o yu4n hui4 huo3 ju4 jie1 li4 ji1ng guo4 shi4 jie4 wu3 da4 zho1u 21 ge4 che2ng shi4
Yah. A.	"Computer, Internet"	"What should I look for when buying a laptop? What is the best brand and what's reliable?","Weight and dimensions are important if you're planning to travel with the laptop. Get something with at least 512 mb of RAM. [...] is a good brand, and has an easy to use site where you can build a custom laptop."

Table 1: Examples of text samples and their labels.

huber, 1997) – there are many works which have shown the ability of LSTMs to model long-range dependencies in NLP applications, e.g. (Sundermeyer et al., 2012; Sutskever et al., 2014) to name just a few. However, we argue that LSTMs are generic learning machines for sequence processing which are lacking task-specific structure.

We propose the following analogy. It is well known that a fully connected one hidden layer neural network can in principle learn any real-valued function, but much better results can be obtained with a deep problem-specific architecture which develops hierarchical representations. By these means, the search space is heavily constrained and efficient solutions can be learned with gradient descent. ConvNets are namely adapted for computer vision because of the compositional structure of an image. Texts have similar properties : characters combine to form n-grams, stems, words, phrase, sentences etc.

We believe that a challenge in NLP is to develop deep architectures which are able to learn hierarchical representations of whole sentences, jointly with the task. In this paper, we propose to use deep architectures of many convolutional layers to approach this goal, using up to 29 layers. The design of our architecture is inspired by recent progress in computer vision, in particular (Simonyan and Zisserman, 2015; He et al., 2016a).

This paper is structured as follows. There have been previous attempts to use ConvNets for text processing. We summarize the previous works in the next section and discuss the relations and differences. Our architecture is described in detail in section 3. We have evaluated our approach on

several sentence classification tasks, initially proposed by (Zhang et al., 2015). These tasks and our experimental results are detailed in section 4. The proposed deep convolutional network shows significantly better results than previous ConvNets approach. The paper concludes with a discussion of future research directions for very deep approach in NLP.

2 Related work

There is a large body of research on sentiment analysis, or more generally on sentence classification tasks. Initial approaches followed the classical two stage scheme of extraction of (hand-crafted) features, followed by a classification stage. Typical features include bag-of-words or n -grams, and their TF-IDF. These techniques have been compared with ConvNets by (Zhang et al., 2015; Zhang and LeCun, 2015). We use the same corpora for our experiments. More recently, words or characters, have been projected into a low-dimensional space, and these embeddings are combined to obtain a fixed size representation of the input sentence, which then serves as input for the classifier. The simplest combination is the element-wise mean. This usually performs badly since all notion of token order is disregarded.

Another class of approaches are recursive neural networks. The main idea is to use an external tool, namely a parser, which specifies the order in which the word embeddings are combined. At each node, the left and right context are combined using weights which are shared for all nodes (Socher et al., 2011). The state of the top node is fed to the classifier. A recurrent neural net-

work (RNN) could be considered as a special case of a recursive NN: the combination is performed sequentially, usually from left to right. The last state of the RNN is used as fixed-sized representation of the sentence, or eventually a combination of all the hidden states.

First works using convolutional neural networks for NLP appeared in (Collobert and Weston, 2008; Collobert et al., 2011). They have been subsequently applied to sentence classification (Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015). We will discuss these techniques in more detail below. If not otherwise stated, all approaches operate on words which are projected into a high-dimensional space.

A rather shallow neural net was proposed in (Kim, 2014): one convolutional layer (using multiple widths and filters) followed by a max pooling layer over time. The final classifier uses one fully connected layer with drop-out. Results are reported on six data sets, in particular Stanford Sentiment Treebank (SST). A similar system was proposed in (Kalchbrenner et al., 2014), but using five convolutional layers. An important difference is also the introduction of multiple *temporal k-max pooling* layers. This allows to detect the k most important features in a sentence, independent of their specific position, preserving their relative order. The value of k depends on the length of the sentence and the position of this layer in the network. (Zhang et al., 2015) were the first to perform sentiment analysis entirely at the character level. Their systems use up to six convolutional layers, followed by three fully connected classification layers. Convolutional kernels of size 3 and 7 are used, as well as simple max-pooling layers. Another interesting aspect of this paper is the introduction of several large-scale data sets for text classification. We use the same experimental setting (see section 4.1). The use of character level information was also proposed by (Dos Santos and Gatti, 2014): all the character embeddings of one word are combined by a max operation and they are then jointly used with the word embedding information in a shallow architecture. In parallel to our work, (Yang et al., 2016) proposed a based hierarchical attention network for document classification that perform an attention first on the sentences in the document, and on the words in the sentence. Their architecture performs very well on datasets whose samples contain multiple sen-

tences.

In the computer vision community, the combination of recurrent and convolutional networks in one architecture has also been investigated, with the goal to “*get the best of both worlds*”, e.g. (Pinheiro and Collobert, 2014). The same idea was recently applied to sentence classification (Xiao and Cho, 2016). A convolutional network with up to five layers is used to learn high-level features which serve as input for an LSTM. The initial motivation of the authors was to obtain the same performance as (Zhang et al., 2015) with networks which have significantly fewer parameters. They report results very close to those of (Zhang et al., 2015) or even outperform ConvNets for some data sets.

In summary, we are not aware of any work that uses VGG-like or ResNet-like architecture to go deeper than than six convolutional layers (Zhang et al., 2015) for sentence classification. Deeper networks were not tried or they were reported to not improve performance. This is in sharp contrast to the current trend in computer vision where significant improvements have been reported using much deeper networks (Krizhevsky et al., 2012), namely 19 layers (Simonyan and Zisserman, 2015), or even up to 152 layers (He et al., 2016a). In the remainder of this paper, we describe our very deep convolutional architecture and report results on the same corpora than (Zhang et al., 2015). We were able to show that performance improves with increased depth, using up to 29 convolutional layers.

3 VDCNN Architecture

The overall architecture of our network is shown in Figure 1. Our model begins with a look-up table that generates a 2D tensor of size (f_0, s) that contain the embeddings of the s characters. s is fixed to 1024, and f_0 can be seen as the “RGB” dimension of the input text.

We first apply one layer of 64 convolutions of size 3, followed by a stack of temporal “convolutional blocks”. Inspired by the philosophy of VGG and ResNets we apply these two design rules: (i) for the same output temporal resolution, the layers have the same number of feature maps, (ii) when the temporal resolution is halved, the number of feature maps is doubled. This helps reduce the memory footprint of the network. The networks contains 3 pooling operations (halving the tempo-

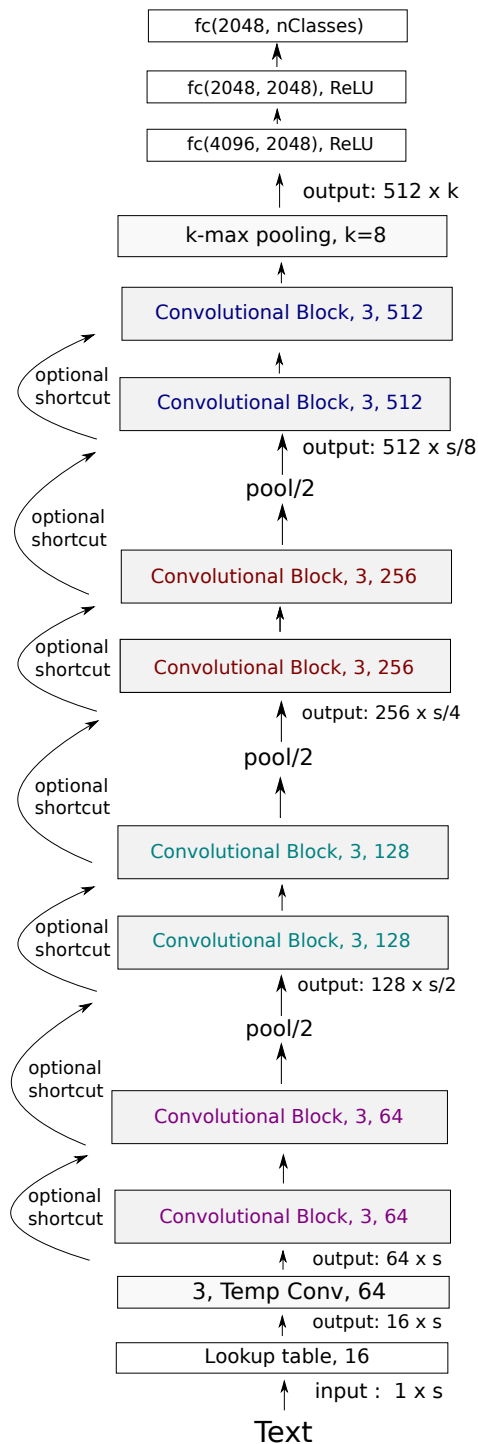


Figure 1: VDCNN architecture.

ral resolution each time by 2), resulting in 3 levels of 128, 256 and 512 feature maps (see Figure 1). The output of these convolutional blocks is a tensor of size $512 \times s_d$, where $s_d = \frac{s}{2^p}$ with $p = 3$ the number of down-sampling operations. At this level of the convolutional network, the resulting tensor can be seen as a high-level representation of the input text. Since we deal with padded input text of fixed size, s_d is constant. However, in the case of variable size input, the convolutional encoder provides a representation of the input text that depends on its initial length s . Representations of a text as a set of vectors of variable size can be valuable namely for neural machine translation, in particular when combined with an attention model. In Figure 1, temporal convolutions with kernel size 3 and X feature maps are denoted "3, Temp Conv, X", fully connected layers which are linear projections (matrix of size $I \times O$) are denoted "fc(I, O)" and "3-max pooling, stride 2" means temporal max-pooling with kernel size 3 and stride 2.

Most of the previous applications of ConvNets to NLP use an architecture which is rather shallow (up to 6 convolutional layers) and combines convolutions of different sizes, e.g. spanning 3, 5 and 7 tokens. This was motivated by the fact that convolutions extract n -gram features over tokens and that different n -gram lengths are needed to model short- and long-span relations. In this work, we propose to create instead an architecture which uses many layers of small convolutions (size 3). Stacking 4 layers of such convolutions results in a span of 9 tokens, but the network can learn by itself how to best combine these different "3-gram features" in a deep hierarchical manner. Our architecture can be in fact seen as a temporal adaptation of the VGG network (Simonyan and Zisserman, 2015). We have also investigated the same kind of "ResNet shortcut" connections as in (He et al., 2016a), namely identity and 1×1 convolutions (see Figure 1).

For the classification tasks in this work, the temporal resolution of the output of the convolution blocks is first down-sampled to a fixed dimension using k -max pooling. By these means, the network extracts the k most important features, independently of the position they appear in the sentence. The $512 \times k$ resulting features are transformed into a single vector which is the input to a three layer fully connected classifier with ReLU

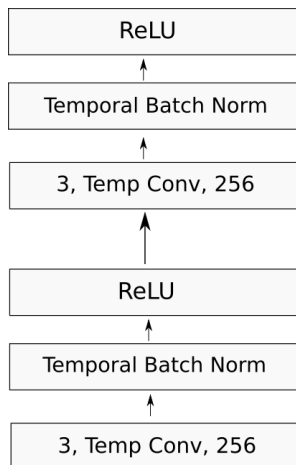


Figure 2: Convolutional block.

hidden units and softmax outputs. The number of output neurons depends on the classification task, the number of hidden units is set to 2048, and k to 8 in all experiments. We do not use drop-out with the fully connected layers, but only temporal batch normalization after convolutional layers to regularize our network.

Convolutional Block

Each convolutional block (see Figure 2) is a sequence of two convolutional layers, each one followed by a temporal BatchNorm (Ioffe and Szegedy, 2015) layer and an ReLU activation. The kernel size of all the temporal convolutions is 3, with padding such that the temporal resolution is preserved (or halved in the case of the convolutional pooling with stride 2, see below). Steadily increasing the depth of the network by adding more convolutional layers is feasible thanks to the limited number of parameters of very small convolutional filters in all layers. Different depths of the overall architecture are obtained by varying the number of convolutional blocks in between the pooling layers (see table 2). Temporal batch normalization applies the same kind of regularization as batch normalization except that the activations in a mini-batch are jointly normalized over temporal (instead of spatial) locations. So, for a mini-batch of size m and feature maps of temporal size s , the sum and the standard deviations related to the BatchNorm algorithm are taken over $|\mathcal{B}| = m \cdot s$ terms.

We explore three types of down-sampling between blocks K_i and K_{i+1} (Figure 1) :

- (i) The first convolutional layer of K_{i+1} has stride 2 (ResNet-like).

Depth:	9	17	29	49
conv block 512	2	4	4	6
conv block 256	2	4	4	10
conv block 128	2	4	10	16
conv block 64	2	4	10	16
First conv. layer	1	1	1	1
#params [in M]	2.2	4.3	4.6	7.8

Table 2: Number of conv. layers per depth.

- (ii) K_i is followed by a k -max pooling layer where k is such that the resolution is halved (Kalchbrenner et al., 2014).

- (iii) K_i is followed by max-pooling with kernel size 3 and stride 2 (VGG-like).

All these types of pooling reduce the temporal resolution by a factor 2. At the final convolutional layer, the resolution is thus s_d .

In this work, we have explored four depths for our networks: 9, 17, 29 and 49, which we define as being the number of convolutional layers. The depth of a network is obtained by summing the number of blocks with 64, 128, 256 and 512 filters, with each block containing two convolutional layers. In Figure 1, the network has 2 blocks of each type, resulting in a depth of $2 \times (2 + 2 + 2 + 2) = 16$. Adding the very first convolutional layer, this sums to a depth of 17 convolutional layers. The depth can thus be increased or decreased by adding or removing convolutional blocks with a certain number of filters. The best configurations we observed for depths 9, 17, 29 and 49 are described in Table 2. We also give the number of parameters of all convolutional layers.

4 Experimental evaluation

4.1 Tasks and data

In the computer vision community, the availability of large data sets for object detection and image classification has fueled the development of new architectures. In particular, this made it possible to compare many different architectures and to show the benefit of very deep convolutional networks. We present our results on eight freely available large-scale data sets introduced by (Zhang et al., 2015) which cover several classification tasks such as sentiment analysis, topic classification or news categorization (see Table 3). The number of training examples varies from 120k up to 3.6M, and the number of classes is comprised between 2

Data set	#Train	#Test	#Classes	Classification Task
AG’s news	120k	7.6k	4	English news categorization
Sogou news	450k	60k	5	Chinese news categorization
DBPedia	560k	70k	14	Ontology classification
Yelp Review Polarity	560k	38k	2	Sentiment analysis
Yelp Review Full	650k	50k	5	Sentiment analysis
Yahoo! Answers	1 400k	60k	10	Topic classification
Amazon Review Full	3 000k	650k	5	Sentiment analysis
Amazon Review Polarity	3 600k	400k	2	Sentiment analysis

Table 3: Large-scale text classification data sets used in our experiments. See (Zhang et al., 2015) for a detailed description.

and 14. This is considerably lower than in computer vision (e.g. 1 000 classes for ImageNet). This has the consequence that each example induces less gradient information which may make it harder to train large architectures. It should be also noted that some of the tasks are very ambiguous, in particular sentiment analysis for which it is difficult to clearly associate fine grained labels. There are equal numbers of examples in each class for both training and test sets. The reader is referred to (Zhang et al., 2015) for more details on the construction of the data sets. Table 4 summarizes the best published results on these corpora we are aware of. We do not use “Thesaurus data augmentation” or any other preprocessing, except lower-casing. Nevertheless, we still outperform the best convolutional neural networks of (Zhang et al., 2015) for all data sets. The main goal of our work is to show that it is possible and beneficial to train very deep convolutional networks as text encoders. Data augmentation may improve our results even further. We will investigate this in future research.

4.2 Common model settings

The following settings have been used in all our experiments. They were found to be best in initial experiments. Following (Zhang et al., 2015), all processing is done at the character level which is the atomic representation of a sentence, same as pixels for images. The dictionary consists of the following characters “abcdefghijklmnopqrstuvwxyz0123456789-.,;.:’”/|_#\$\$%^&*~\`+=<>()[]{}” plus a special padding, space and unknown token which add up to a total of 69 tokens. The input text is padded to a fixed size of 1014, larger text are truncated. The character embedding is

of size 16. Training is performed with SGD, using a mini-batch of size 128, an initial learning rate of 0.01 and momentum of 0.9. We follow the same training procedure as in (Zhang et al., 2015). We initialize our convolutional layers following (He et al., 2015). One epoch took from 24 minutes to 2h45 for depth 9, and from 50 minutes to 7h (on the largest datasets) for depth 29. It took between 10 to 15 epoches to converge. The implementation is done using Torch 7. All experiments are performed on a single NVidia K40 GPU. Unlike previous research on the use of ConvNets for text processing, we use temporal batch norm without dropout.

4.3 Experimental results

In this section, we evaluate several configurations of our model, namely three different depths and three different pooling types (see Section 3). Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with small temporal convolution filters with different types of pooling, which shows that a significant improvement on the state-of-the-art configurations can be achieved on text classification tasks by pushing the depth to 29 convolutional layers.

Our deep architecture works well on big data sets in particular, even for small depths. Table 5 shows the test errors for depths 9, 17 and 29 and for each type of pooling : convolution with stride 2, k -max pooling and temporal max-pooling. For the smallest depth we use (9 convolutional layers), we see that our model already performs better than Zhang’s convolutional baselines (which includes 6 convolutional layers and has a different architecture) on the biggest data sets : Yelp Full, Yahoo Answers and Amazon Full and Polarity. The most important decrease in classification error can

Corpus:	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Method	n-TFIDF	n-TFIDF	n-TFIDF	ngrams	Conv	Conv+RNN	Conv	Conv
Author	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Xiao]	[Zhang]	[Zhang]
Error	7.64	2.81	1.31	4.36	37.95*	28.26	40.43*	4.93*
[Yang]	-	-	-	-	-	24.2	36.4	-

Table 4: Best published results from previous work. Zhang et al. (2015) best results use a Thesaurus data augmentation technique (marked with an *). Yang et al. (2016)’s hierarchical methods is particularly adapted to datasets whose samples contain multiple sentences.

Depth	Pooling	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
9	Convolution	10.17	4.22	1.64	5.01	37.63	28.10	38.52	4.94
9	KMaxPooling	9.83	3.58	1.56	5.27	38.04	28.24	39.19	5.69
9	MaxPooling	9.17	3.70	1.35	4.88	36.73	27.60	37.95	4.70
17	Convolution	9.29	3.94	1.42	4.96	36.10	27.35	37.50	4.53
17	KMaxPooling	9.39	3.51	1.61	5.05	37.41	28.25	38.81	5.43
17	MaxPooling	8.88	3.54	1.40	4.50	36.07	27.51	37.39	4.41
29	Convolution	9.36	3.61	1.36	4.35	35.28	27.17	37.58	4.28
29	KMaxPooling	8.67	3.18	1.41	4.63	37.00	27.16	38.39	4.94
29	MaxPooling	8.73	3.36	1.29	4.28	35.74	26.57	37.00	4.31

Table 5: Testing error of our models on the 8 data sets. No data preprocessing or augmentation is used.

be observed on the largest data set Amazon Full which has more than 3 Million training samples. We also observe that for a small depth, temporal max-pooling works best on all data sets.

Depth improves performance. As we increase the network depth to 17 and 29, the test errors decrease on all data sets, for all types of pooling (with 2 exceptions for 48 comparisons). Going from depth 9 to 17 and 29 for Amazon Full reduces the error rate by 1% absolute. Since the test is composed of 650K samples, 6.5K more test samples have been classified correctly. These improvements, especially on large data sets, are significant and show that increasing the depth is useful for text processing. Overall, compared to previous state-of-the-art, our best architecture with depth 29 and max-pooling has a test error of 37.0 compared to 40.43%. This represents a gain of 3.43% absolute accuracy. The significant improvements which we obtain on all data sets compared to Zhang’s convolutional models do not include any data augmentation technique.

Max-pooling performs better than other pooling types. In terms of pooling, we can also see that max-pooling performs best overall, very close to convolutions with stride 2, but both are significantly superior to k -max pooling.

Both pooling mechanisms perform a max operation which is local and limited to three consec-

utive tokens, while k -max pooling considers the whole sentence at once. According to our experiments, it seems to hurt performance to perform this type of max operation at intermediate layers (with the exception of the smallest data sets).

Our models outperform state-of-the-art ConvNets. We obtain state-of-the-art results for all data sets, except AG’s news and Sogou news which are the smallest ones. However, with our very deep architecture, we get closer to the state-of-the-art which are ngrams TF-IDF for these data sets and significantly surpass convolutional models presented in (Zhang et al., 2015). As observed in previous work, differences in accuracy between shallow (TF-IDF) and deep (convolutional) models are more significant on large data sets, but we still perform well on small data sets while getting closer to the non convolutional state-of-the-art results on small data sets. The very deep models even perform as well as ngrams and ngrams-TF-IDF respectively on the sentiment analysis task of Yelp Review Polarity and the ontology classification task of the DBpedia data set. Results of Yang et al. (only on Yahoo Answers and Amazon Full) outperform our model on the Yahoo Answers dataset, which is probably linked to the fact that their model is task-specific to datasets whose samples that contain multiple sentences like (question, answer). They use a hierarchical attention mecha-

nism that apply very well to documents (with multiple sentences).

Going even deeper degrades accuracy. Shortcut connections help reduce the degradation.

As described in (He et al., 2016a), the gain in accuracy due to the the increase of the depth is limited when using standard ConvNets. When the depth increases too much, the accuracy of the model gets saturated and starts degrading rapidly. This *degradation* problem was attributed to the fact that very deep models are harder to optimize. The gradients which are backpropagated through the very deep networks vanish and SGD with momentum is not able to converge to a correct minimum of the loss function. To overcome this degradation of the model, the *ResNet model* introduced shortcut connections between convolutional blocks that allow the gradients to flow more easily in the network (He et al., 2016a).

We evaluate the impact of shortcut connections by increasing the number of convolutions to 49 layers. We present an adaptation of the ResNet model to the case of temporal convolutions for text (see Figure 1). Table 6 shows the evolution of the test errors on the Yelp Review Full data set with or without shortcut connections. When looking at the column “without shortcut”, we observe the same degradation problem as in the original ResNet article: when going from 29 to 49 layers, the test error rate increases from 35.28 to 37.41 (while the training error goes up from 29.57 to 35.54). When using shortcut connections, we observe improved results when the network has 49 layers: both the training and test errors go down and the network is less prone to underfitting than it was without shortcut connections.

While shortcut connections give better results when the network is very deep (49 layers), we were not able to reach state-of-the-art results with them. We plan to further explore adaptations of residual networks to temporal convolutions as we think this a milestone for going deeper in NLP. Residual units (He et al., 2016a) better adapted to the text processing task may help for training even deeper models for text processing, and is left for future research.

Exploring these models on text classification tasks with more classes sounds promising.

Note that one of the most important difference between the classification tasks discussed in this

depth	without shortcut	with shortcut
9	37.63	40.27
17	36.10	39.18
29	35.28	36.01
49	37.41	36.15

Table 6: Test error on the Yelp Full data set for all depths, with or without residual connections.

work and ImageNet is that the latter deals with 1000 classes and thus much more information is back-propagated to the network through the gradients. Exploring the impact of the depth of temporal convolutional models on categorization tasks with hundreds or thousands of classes would be an interesting challenge and is left for future research.

5 Conclusion

We have presented a new architecture for NLP which follows two design principles: 1) operate at the lowest atomic representation of text, i.e. characters, and 2) use a deep stack of local operations, i.e. convolutions and max-pooling of size 3, to learn a high-level hierarchical representation of a sentence. This architecture has been evaluated on eight freely available large-scale data sets and we were able to show that increasing the depth up to 29 convolutional layers steadily improves performance. Our models are much deeper than previously published convolutional neural networks and they outperform those approaches on all data sets. To the best of our knowledge, this is the first time that the “*benefit of depths*” was shown for convolutional neural networks in NLP.

Eventhough text follows human-defined rules and images can be seen as raw signals of our environment, images and small texts have similar properties. Texts are also compositional for many languages. Characters combine to form n-grams, stems, words, phrase, sentences etc. These similar properties make the comparison between computer vision and natural language processing very profitable and we believe future research should invest into making text processing models deeper. Our work is a first attempt towards this goal.

In this paper, we focus on the use of very deep convolutional neural networks for sentence classification tasks. Applying similar ideas to other sequence processing tasks, in particular neural machine translation is left for future research. It needs to be investigated whether these also benefit from having deeper convolutional encoders.

References

- Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *NIPS*, volume 13, pages 932–938, Vancouver, British Columbia, Canada.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, Helsinki, Finland.
- Ronan Collobert, Jason Weston, Léon Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, pages 2493–2537.
- Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78, Dublin, Ireland.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, Santiago, Chile.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, Nevada, USA.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645, Amsterdam, Netherlands. Springer.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, Lille, France.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, Baltimore, Maryland, USA.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, Lake Tahoe, California, USA.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.
- Pedro HO Pinheiro and Ronan Collobert. 2014. Recurrent convolutional neural networks for scene labeling. In *ICML*, pages 82–90, Beijing, China.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. pages 1715–1725.
- Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*, San Diego, California, USA.
- Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161, Edinburgh, UK. Association for Computational Linguistics.
- Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, Portland, Oregon, USA.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, Montreal, Canada.
- Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489, San Diego, California, USA.
- Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833, Zurich, Switzerland. Springer.
- Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657, Montreal, Canada.

“PageRank” for Argument Relevance

Henning Wachsmuth and Benno Stein and Yamen Ajjour

Faculty of Media, Bauhaus-Universität Weimar, Germany

{henning.wachsmuth, benno.stein, yamen.ajjour}@uni-weimar.de

Abstract

Future search engines are expected to deliver pro and con arguments in response to queries on controversial topics. While argument mining is now in the focus of research, the question of how to retrieve the *relevant* arguments remains open. This paper proposes a radical model to assess relevance objectively at web scale: the relevance of an argument’s conclusion is decided by what other arguments reuse it as a premise. We build an argument graph for this model that we analyze with a recursive weighting scheme, adapting key ideas of PageRank. In experiments on a large ground-truth argument graph, the resulting relevance scores correlate with human average judgments. We outline what natural language challenges must be faced at web scale in order to stepwise bring argument relevance to web search engines.

1 Introduction

What stance should I take? What are the best arguments to back up my stance? Information needs of people aim more and more at arguments in favor of or against a given controversial topic (Cabrio and Villata, 2012b). As a result, future information systems, above all search engines, are expected to deliver pros and cons in response to respective queries (Rinott et al., 2015). Recently, argument mining has become emerging in research, also being studied for the web (Al-Khatib et al., 2016a). Such mining finds and relates units of arguments (i.e., premises and conclusions) in natural language text, but it does not assess what arguments are *relevant* for a topic. Consider the following arguments (with implicit conclusions) for a query “reasons against capital punishment”:

Example a_1 . “*The death penalty legitimizes an irreversible act of violence. As long as human justice remains fallible, the risk of executing the innocent can never be eliminated.*”

Example a_2 . “*Capital punishment produces an unacceptable link between the law and violence.*”¹

While both arguments are on-topic, a_1 seems more clear, concrete, and targeted, potentially making it more relevant. First approaches to assess argument quality exist (see Section 2). However, they hardly account for the problem that argument quality (and relevance in particular) is often perceived subjectively. Whether a_3 , e.g., is more relevant than a_1 or less depends on personal judgment:

Example a_3 . “*The death penalty doesn’t deter people from committing serious violent crimes. The thing that deters is the likelihood of being caught and punished.*”

In this paper, we study from a retrieval perspective how to assess argument relevance *objectively*, i.e., without relying on explicit human judgments. Following argumentation theory, we see relevance as a dialectical quality that depends on how beneficial all participants of a discussion deem the use of an argument for the discussion (Walton, 2006). In the context of web search, an objective assessment hence at best takes place at web scale. We propose the radical model that relevance is not decided by the content of arguments, but structurally by how many arguments across the web use the conclusion of an argument as a premise and by how relevant these are in turn. The rationale is that an author cannot control who “cites” his or her argument in this way, so each citation can be assumed to add to relevance. Thereby, we achieve to decouple relevance from the soundness of the inference an argument makes to draw its conclusion.

¹All example arguments in Sections 1–4 are derived from www.bbc.co.uk/ethics/capitalpunishment/against_1.shtml.

Given algorithms that mine arguments from the web and that decide if two argument units mean the same, an argument graph can be built where nodes represent arguments and each edge the reuse of a conclusion as a premise (Section 3). Based on the graph, we devise an adaptation of the PageRank algorithm (Page et al., 1999) to assess argument relevance. Originally, PageRank recursively processes the web link graph to infer the objective relevance of each web page from what other pages link to that page. In an according manner, we process the argument graph to compute a score for each argument unit. An argument’s relevance then follows from the scores of its premises (Section 4). Analogue to the supportive nature of web links, our new *PageRank for argument relevance* counts any use of a conclusion as a premise as a support of relevance. In principle, balancing support and attack relations would also be possible, though.

At web scale, mining arguments from natural language text raises complex challenges. Since not all have been solved reliably yet, we here derive an argument graph from the complete *Argument Web* (Bex et al., 2013), a large ground-truth database consisting of about 50,000 argument units. This way, we can evaluate PageRank without the noise induced by mining errors. Moreover, we provide a first argument relevance benchmark dataset, where seven experts ranked arguments for 32 conclusions of general interest (Section 5). On the dataset, the PageRank scores beat several intuitive baselines and correlate with human average judgments of relevance—even though they ignore an argument’s content and inference—indicating the impact of our approach (Section 6). We discuss how to bring argument relevance to web search engines, starting from the technologies of today (Section 7).

Contributions To summarize, the work at hand provides three main contributions to research:

1. An approach to structurally and hence objectively assess argument relevance at web scale.
2. A first benchmark ranking dataset for the evaluation of argument relevance assessment.
3. Evidence that argument relevance depends on the reuse of conclusions in other arguments.

2 Related Work

Argument relevance can be seen as one dimension of argumentation quality. In argumentation theory, two relevance types are distinguished: *Local* rele-

vance means that an argument’s premises actually help accepting or rejecting its conclusion. Such relevance is one prerequisite of a cogent argument, along with the acceptability of the premises and their sufficiency for drawing the conclusion (Johnson and Blair, 2006). Here, we are interested in an argument’s *global* relevance, which refers to the benefit of the argument in a discussion (Walton, 2006): An argument is more globally relevant the more it contributes to resolving an issue (van Eemeren, 2015). While Blair (2012) deems both types as vague and resisting analysis so far, we assess global relevance using objective statistics.

In (Wachsmuth et al., 2017), we comprehensively survey theories on argumentation quality as well as computational approaches to specific quality dimensions. Among the latter, Persing and Ng (2015) rely on manual annotations of essays to predict how strong an essay’s argument is—a naturally subjective and non-scalable assessment. For scalability, Habernal and Gurevych (2016) learn on crowdsourced labels, which of two arguments is more convincing. Similar to us, they construct a graph to rank arguments, but since their graph is based on the labels, the subjectivity remains. This also holds for (Braunstein et al., 2016) where classical retrieval and argument-related features serve to rank argument units by the level of support they provide in community question answering.

More objectively, Boltužić and Šnajder (2015) find popular arguments in online debates. However, popularity alone is often not correlated with merit (Govier, 2010). We additionally analyze dependencies between arguments—like Cabrio and Villata (2012a) who classify attack relations between debate portal arguments. From these, they derive accepted arguments in the logical argumentation framework of Dung (1995). Relevance and acceptability are orthogonal dimensions: an argument may be relevant even if far from everyone accepts it. While probabilistic extensions of Dung’s framework exist (Bistarelli et al., 2011; Dondio, 2014), they aim at the probability of logical truth. In contrast, relevance reflects the importance of arguments, for which we take on a retrieval view.

In information retrieval, relevance represents a fundamental concept, particularly in the context of search engines. A web page is seen as relevant for a search query if it contains information the querying person was looking for (Croft et al., 2009). To assess argument relevance objectively, we adapt

a core retrieval technique, recursive link analysis. Due to its wide use, we build upon Google’s original PageRank algorithm (Page et al., 1999), but alternatives such as (Kleinberg, 1999) would also apply. PageRank is sensitive to certain manipulations, such as link farms (Croft et al., 2009). Some of them will affect argument graphs, too. Improvements of the original algorithm should therefore be taken into account in future work. In this paper, we omit them on purpose for simplicity and clarity.

We already introduced our PageRank approach in (Al-Khatib et al., 2016a), but we only roughly sketched its general idea there. Recursive analyses have also been proposed for fact finding, assuming that trustworthy web pages contain many true facts, and that true facts will be found on many trustworthy web pages (Yin et al., 2007; Galland et al., 2010). Pasternack and Roth (2010) model a user’s prior knowledge in addition. Close to argumentation, Samadi et al. (2016) evaluate claims using a credibility graph derived from evidence found in web pages. All these works target truth. In order to capture relevance, we base PageRank on the reuse of argument units instead.

In particular, we construct a graph from all arguments found in web pages. Both complex argument models from theory (Toulmin, 1958; Reisert et al., 2015) and simple proprietary models (Levy et al., 2014) have been studied for web text. Some include quality-related concepts, such as evidence types (Al-Khatib et al., 2016b). Others represent the overall structure of argumentation (Wachsmuth et al., 2015). Like Mochales and Moens (2011), we consider only premises and conclusions as units of arguments here. This is the common ground of nearly all argument-level models, and it will allow an integration with approaches to analyze argument inference (Feng and Hirst, 2011) based on the argumentation schemes of Walton et al. (2008). Edges in our graph emerge from the usage of units in different arguments. Alternatively, it would be possible to mine support and attack relations between arguments (Park and Cardie, 2014; Peldszus and Stede, 2015).

Not all mining steps work robustly on web text yet (Al-Khatib et al., 2016a). To focus on the impact of PageRank, we thus rely on ground-truth data in our experiments. In isolation, existing argument corpora do not adequately mimic web context, as they are small and dedicated to a specific genre (Stab and Gurevych, 2014), or restricted to

flat relations between units (Aharoni et al., 2014). To maximize size and heterogeneity, we here refer to the Argument Web (Bex et al., 2013), which is to our knowledge the largest ground-truth argument database available so far. It includes relation-rich corpora, e.g., AraucariaDB (Reed and Rowe, 2004), as well as much annotated web text, e.g., from (Walker et al., 2012) and (Wacholder et al., 2014). Thus, it serves as a suitable basis for constructing an argument graph.

3 The Web as an Argument Graph

We now present the model that we envision as the basis for argument relevance in future web search, targeting information needs of the following kind: “*What are the most relevant arguments to support or attack my stance?*” The model relies on three principles that aim at the separation of concerns:

- I. *Freedom of Inference.* No inference from argument premises to conclusions is challenged.
- II. *Freedom of Mining.* No restrictions are made for how to mine and relate argument units.
- III. *Freedom of Assessment.* No graph processing method is presupposed to assess relevance.²

3.1 Definition of the Argument Graph

Let $D = \{d_1, d_2, \dots\}$ be the set of all considered web pages. Each $d \in D$ may contain zero or more arguments. Given D , we model the web as an argument graph in three incremental building blocks:

Canonical Argument Structure A canonical structure that represents each argument in D as a tuple $a = \langle c, P \rangle$ where c denotes the conclusion of a and $P = \{c_1, \dots, c_k\}$ its premises, $k \geq 0$. Both conclusions and premises form argument units.

Reuse Interpretation Function An interpretation function \mathcal{I} that assigns one label from the set $\{\approx, \not\approx\}$ to each pair of argument units (c, c') from all arguments in D .

Argument Graph A graph $G = (A, E)$ such that

$A = \{a_1, \dots, a_n\}$ is a set of nodes where each $a \in A$ corresponds to one argument in D ;

$E \subseteq A \times A$ is a set of edges where $(a, a') \in E$ iff. $\mathcal{I}(c, c_i) = \approx$ holds for the conclusion c of a and any premise c_i of a' .³

²We will introduce a specific method in Section 4. Nevertheless, other methods would also be applicable in principle.

³In order to keep the definition of the argument graph simple, we include *all* possible pairs of arguments for the edges here. In practice, some pairs should rather be excluded in order to counter manipulation, e.g., those *within* a web page.

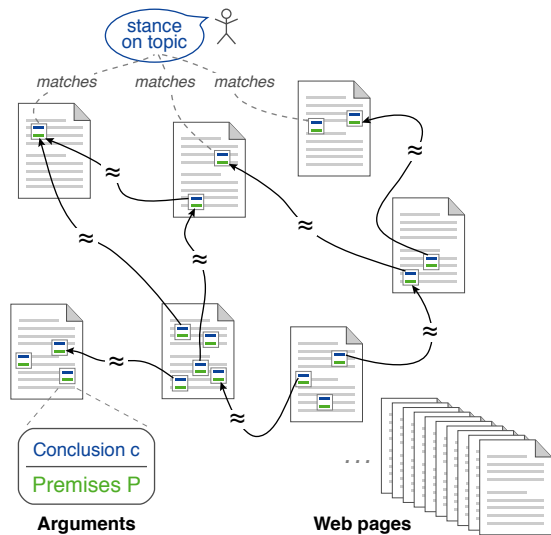


Figure 1: A small argument graph with three potentially relevant arguments for a queried stance.

Figure 1 sketches an argument graph. Given a user query with a stance on a controversial topic, as shown, each argument whose (maybe implicit) conclusion c matches the stance is potentially relevant. Stance classification is outside the scope of this paper. We assess the relevance of arguments with conclusion c . The reuse of such conclusions in other arguments is exemplified in Figure 2.

3.2 Properties of the Argument Graph Model

In accordance with Principle I, the canonical structure implicitly accepts the inference that an argument draws to arrive at its conclusion. This separates soundness from relevance, reducing the latter to an argument’s units. We even permit “arguments” that have no premise. The reason is that argument units can be relevant without justification (e.g., when serving as axioms for others).

In accordance with Principle II, we do not detail the semantics of the concepts that we propose to construct arguments and their relations, leaving the exact interpretation to the mining algorithms at hand. For arguments, premises and conclusions denote the common ground, and they are generally identifiable in various web texts. For relations, the definition based on the reuse of argument units actually refines previous rather vague relation models, such as (Dung, 1995)—this is possible due to the abstraction from inference.

In accordance with Principle III, we do not pre-define how to assess relevance given an argument graph (and the web pages). In addition to a conclu-

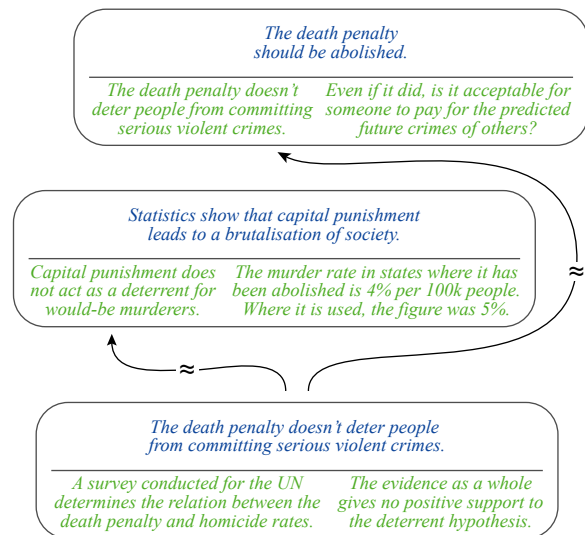


Figure 2: Example for the reuse of an argument’s conclusion as a premise in two other arguments.

sion, e.g., its opposite can be generated (Bilu et al., 2015) to balance support and attack somehow. In general, the usage of conclusions as premises favors a monotonous assessment (the more the better), which we implement in Section 4. Note that we allow circles in the graph. This might look unwanted as it enables circular reasoning. However, not all arguments use the same inference rule (say, modus ponens). Hence, it is reasonable that they, directly or indirectly, refer to each other.

Altogether, our model defines a *framework* for assessing argument relevance. It is instantiated by concrete mining and graph processing algorithms. An analysis of argument inference should complement this, e.g., to filter out unsound arguments. Despite its framework nature, the model suggests a recursive assessment where an argument is more relevant the more relevant arguments it relates to.

4 PageRank for Argument Relevance

Given an argument graph $G = (A, E)$, we propose to assess argument relevance structurally and thus objectively. In the following, we first develop how to adapt PageRank in order to recursively compute relevance scores for all units of the arguments in A based on E . Then, we discuss how to derive the relevance of each argument from these scores.

4.1 PageRank for Conclusion Relevance

PageRank revolutionized web search, because it introduced “a method for rating web pages objectively and mechanically, effectively measuring the

human interest and attention” (Page et al., 1999). The original method assigns a high PageRank $p(d)$ to a web page d if d is linked by many other web pages with a high PageRank. This value corresponds to the probability that a web surfer, who either follows a link on a visited web page or randomly chooses a new page, enters d . In particular, based on the link graph induced by a set of web pages D , $p(d)$ is computed recursively as:

$$p(d) = (1 - \alpha) \cdot \frac{1}{|D|} + \alpha \cdot \sum_i \frac{p(d_i)}{|D_i|}$$

Here, $p(d_i)$ is the PageRank score of a web page $d_i \in D$ that links to d , and D_i is the set of all web pages that d_i links to. According to the right summand, a web page linking to d contributes to $p(d)$ more the less outgoing links it has, in order to reward the focus on specific links. The left summand specifies an equal ground relevance $\frac{1}{|D|}$ for all web pages d , summing up to 1. The factor $\alpha \in [0, 1]$ weights the two summands.

Based on an argument graph $G = (A, E)$, we adapt the PageRank idea in order to analogously rate the conclusion c of each argument $a \in A$ “objectively and mechanically”. Recall that an edge $(a, a') \in E$ states that c is a premise in another argument $a' \in A$. Now, we assign a high PageRank $\hat{p}(c)$ to c if c serves as a premise for many conclusions c_i with high $\hat{p}(c_i)$. For this, we adjust the equation above in two ways:

Ground Relevance Originally, PageRank works on the lowest layer of the web, the link graph. This layer has no specific entry point, which is why the ground relevance of all pages $d \in D$ is the same in $p(d)$ above. Working with arguments on web pages, however, adds a new layer on top. Therefore, we start with the original PageRank as the ground relevance, i.e., we postulate that the higher $p(d)$ is, the more relevant is a conclusion c found on d by default. In order to maintain a sum of 1 for all arguments, we normalize $p(d)$ with the average number of arguments per web page. This results in the ground relevance $\frac{p(d) \cdot |D|}{|A|}$ for each c .

Recursive Relevance The author of a web page d can specify the web pages that d links to, but the author cannot control which pages eventually link to d . This contrast is a cornerstone of the original PageRank to model relevance objectively. By analogy, the author of an argument specifies which argument units to use as premises for the argument’s conclusion c , but the author cannot control

which arguments use c as a premise for their conclusion c_i . This contrast is a cornerstone of our “PageRank for arguments” to model relevance objectively. In order to reward a focus on specific conclusions, we normalize the impact of the relevance $\hat{p}(c_i)$ of each conclusion c_i , for which c serves as a premise, on the relevance of c by the number of premises $|P_i|$ given for c_i . This results in the contribution $\frac{\hat{p}(c_i)}{|P_i|}$ for each c_i .

Altogether, we compute the PageRank of a conclusion c that is contained in a web page d as:

$$\hat{p}(c) = (1 - \alpha) \cdot \frac{p(d) \cdot |D|}{|A|} + \alpha \cdot \sum_i \frac{\hat{p}(c_i)}{|P_i|}$$

4.2 Properties of the PageRank Approach

For space reasons, we only sketch that the adapted PageRank $\hat{p}(c)$ maintains two important properties of the original PageRank (Page et al., 1999).

First, by construction, the original scores $p(d)$ of all web pages sum up to 1. The left summand of $\hat{p}(c)$ shares this sum among all arguments. The right summand ensures that the total contribution of conclusion usages is normalized with the total number of premises. Thus, the sum of all adapted PageRank scores is also 1.

Second, as the original PageRank, $\hat{p}(c)$ reflects the idea of a citation ranking: Basic conclusions that serve as a premise for many arguments get a high score. They take the role of fundamental literature, say, “*human life is valuable*” in the context of death penalty. At the other end, each conclusion of a leaf argument in the graph is assigned only its ground relevance, since it is never reused. Without citations, relevance can still be estimated based on authorship, e.g., finding an argument on the BBC page from Footnote 1 might suffice to deem it relevant. We model this by including $p(d)$ in $\hat{p}(c)$.

4.3 From Conclusion to Argument Relevance

Given a conclusion c , all arguments $\langle c, P \rangle$ compete in terms of relevance. Since each such argument has the same conclusion, its relevance needs to be derived from its premises P . Intuitively, an argument proves only as strong at its weakest premise, so the minimum premise PageRank score could govern relevance. This fits our model, as we have “outsourced” the soundness of the inference based on the premises. However, it favors arguments with few premises. In order to find the best derivation, we compare four different premise aggregation methods in Section 6:

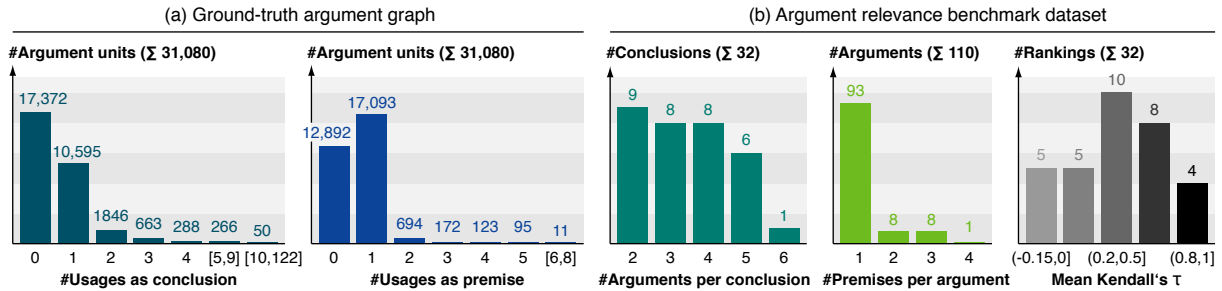


Figure 3: (a) Histograms of the usages of all argument units in the ground-truth argument graph as a conclusion or premise, respectively. (b) Histograms of the arguments per conclusion and the premises per argument in the benchmark dataset as well as the mean Kendall's τ rank correlation of all seven rankers.

- Minimum.* The relevance of an argument corresponds to its minimum premise PageRank.
- Average.* The relevance of an argument corresponds to its average premise PageRank.
- Maximum.* The relevance of an argument corresponds to its maximum premise PageRank.
- Sum.* The relevance of an argument corresponds to the sum of its premise PageRanks.

In general, as for web pages (Croft et al., 2009), PageRank should certainly not be seen as the ultimate way of assessing argument relevance, especially because it fully ignores the content and inference of arguments. Rather, it provides an objective means to identify arguments commonly referred to for a given conclusion.

5 The Webis-ArgRank-17 Dataset

This section describes our construction of a large ground-truth argument graph as well as our creation of manual relevance rankings of arguments from the graph. The resulting *Webis-ArgRank-17 dataset* is not meant for training statistical ranking approaches. Rather, it serves as a first benchmark for evaluating argument relevance assessment.⁴

5.1 A Large Ground-Truth Argument Graph

As discussed in Section 2, the *Argument Web* is the largest existing argument database. It contains structured argument corpora (several from published research) with diverse types of mostly English text, often web content.⁵ The *Argument Web* stores annotations in a standard format, so called argument maps. Each map specifies nodes that

correspond to argument units or to inference rules. Edges connect one of each in either direction. Implicitly, incoming edges of an inference node define premises of an argument, the single outgoing edge an argument's conclusion. At the last date we accessed the *Argument Web* (June 2, 2016), it contained 57 corpora with 8479 maps, summing up to 49,504 argument units and 26,012 arguments.

In order to get a ground-truth argument graph of maximum size, we merged all argument maps except for duplicates. We created one argument node for each inference node while maintaining argument units not connected to any inference node for completeness. For the edges of the argument graph, we assumed two units to be the same if and only if they capture exactly the same text, thereby minimizing the number of falsely detected usages of conclusions. Figure 3(a) shows how many units are used how often as a premise and as a conclusion respectively.⁶

The constructed graph contains 31,080 different argument units, 28,795 of which participate in 17,877 arguments. For convenience, we already precomputed the adapted PageRank score $\hat{p}(c)$ of each argument unit c as well as the frequency of c in the graph. As no original PageRank score $p(d)$ can be accessed for c , we started with the same ground relevance $\frac{1}{31,080}$ for all units.

5.2 Benchmark Argument Rankings

3113 conclusions in the constructed graph have more than one argument and, so, are candidates for ranking. From these, we selected all 498 conclusions for which at least one argument has multiply

⁴The dataset and the Java code for reproducing all experiment results are freely available at: <http://www.arguana.com>

⁵All corpora contained in the *Argument Web* can be found at: <http://www.arg.dundee.ac.uk/aif-corpora>

⁶We tested some high-precision heuristics to match units that occur multiple times in different manifestations, such as ignoring capitalization or discourse connectives. However, the effect was little, which is why we decided to stick with exact matches to avoid false positives in the ground-truth graph.

Conclusion	Argument	Premises of Argument	Rank	\emptyset
Strawberries are the best choice for your breakfast meal!	a_1 (pro)	“Berries are superfoods because they’re so high in antioxidants without being high in calories”, says Giovinazzo _[premise 1] MS, RD, a nutritionist at Clay health club and spa, in New York City. _[premise 2]	1	1.43
	a_2 (pro)	One cup of strawberries, for instance, contains your full recommended daily intake of vitamin C, along with high quantities of folic acid and fiber.	2	1.57
	a_3 (pro)	Strawberries are good for your ticker.	3	3.00
Technology has enhanced the daily life of humans.	a_4 (pro)	The internet has enabled us to widen our knowledge.	1	2.00
	a_5 (pro)	Technology has given us a means of social interaction that wasn’t possible before.	2	2.71
	a_6 (pro)	The use of technology has revolutionized business.	3	3.14
	a_7 (con)	No longer is shopping a personal experience, you’re mostly dealing with computers when you’re purchasing online.	4	3.43
	a_8 (con)	Social interactions via the internet are a huge waste of time.	5	4.29
	a_9 (con)	There’s a ton of information on the internet that is entirely useless.	6	5.42

Table 1: Two argument conclusions in the benchmark dataset, together with the premises of all alternative pro and con arguments, the arguments’ ranks in the dataset, and the mean ranks assigned by the 7 rankers.

used premises, as all others show no structural difference in the graph. We then let two experts from computational linguistics classify for each conclusion as to whether it denotes (a) a claim that internet users might search arguments for or (b) not such a claim for any of five reasons: (1) It is not of general interest but comes from a personal or any other too specific context, e.g., “*Viv needs to be allowed to prove herself*”, (2) its meaning is unclear, e.g., “*we need to get back to the classics*”, (3) it is not in English, (4) it mixes multiple conclusions, or (5) it is not a real conclusion but a topic, anecdote, question, or description, e.g., “*fingerprinting at the airport*” or “*what!?*”.

The experts could access the premises to see if unclear references can be resolved. They chose the same class 451 times (90.6%) with a substantial Cohen’s κ agreement of 0.69. In 136 cases, no expert saw a real claim, indicating some noise in the data. For the rankings, we selected only those conclusions that both saw as claims. We disregarded multiple instances of an argument and the few conclusions where only one argument was left then.

Next, each of the 264 arguments for the remaining 70 conclusions was classified by the same experts as to whether it is (a) a correct argument for the conclusion, (b) a correct counterargument, or (c) not correct for lack of real premises. Restatements of the conclusion as well as ad-hominem attacks were not seen as premises, while the experts were asked to ignore an argument’s strength.

The experts agreed in 201 cases (76.1%) with $\kappa = 0.63$. An example that they saw differently is “*I agree... the thrill is gone*” for the conclusion “*a tweet is fundamentally valueless*”. To al-

low for a reasonable but tractable ranking, we kept only conclusions where the experts agreed on two to six arguments and/or counterarguments, and we discarded one conclusion that paraphrased another one. The resulting dataset covers 32 conclusions. We included all 110 arguments for these conclusions, since their relevance is assessed via ranking. Figure 3(b) shows the distribution of arguments over conclusions and the premises per argument. For two conclusions, all premises are listed in Table 1. The ranks resulted from the final step.

In particular, since we expected argument relevance to be perceived subjectively, a total of seven experts from computational linguistics and information retrieval ranked all arguments (in terms of a strict ordering) for each conclusion by how much they contribute to the acceptance or rejection of the conclusion. In order not to bias the experts, they received arguments with corrected grammar, resolved references, and merged premises. They should follow their own view but acknowledge that there may be relevant counterarguments.

The highest agreement of two experts on all 32 rankings was 0.59 in terms of Kendall’s τ rank correlation (Pearson coefficient 0.63), the mean over all expert pairs 0.36 (Pearson 0.40). This gap supports the subjectivity hypothesis. Figure 3(b) shows that five rankings had a negative τ -value for all experts (the lowest τ was -0.14), whereas in 12 cases τ was above 0.5, in four cases above 0.8. Overall, the resulting ranks thus largely qualify as benchmark average relevance judgments.⁷

⁷We are aware that the seven chosen experts are certainly not representative of average web users. In order to achieve a controlled setting, however, we preferred to rely on experts, e.g., to avoid misconceptions of terms such as “argument”.

# Approach	(a) Minimum			(b) Average			(c) Maximum			(d) Sum			(e) Best results		
	τ	best	worst	τ	best	worst	τ	best	worst	τ	best	worst	τ	best	worst
1 PageRank	0.01	11	6	0.02	12	6	0.11	11	4	0.28	15	3	0.28	15	3
2 Frequency	-0.10	5	9	-0.03	8	10	-0.01	7	9	0.10	11	9	0.10	11	9
3 Similarity	-0.13	7	12	-0.05	8	11	0.01	9	10	0.02	9	10	0.02	9	10
4 Sentiment	0.01	8	6	0.11	12	4	0.12	10	5	0.12	12	4	0.12	12	4
5 Most premises	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.19	6	1
6 Random	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	0.00	8	7

Table 2: (a–d) Mean Kendall’s τ correlation of each approach with all benchmark argument rankings and counts of the *best* / *worst* rankings, once for each premise aggregation method. (e) Best observed results.

6 Evaluation

Finally, we report on an experiment that we carried out on the whole argument graph from Section 5 in order to provide first evidence that our PageRank approach objectively assesses argument relevance. Here, we assume that the average judgments of our benchmark rankings reflect objective relevance.

Approaches We compare six ranking approaches below. In case of 1.–4., we evaluate all premise aggregation methods from Section 4: (a) *Minimum*, (b) *Average*, (c) *Maximum*, and (d) *Sum*. While we are aware that more sophisticated ranking approaches are possible, the considered selection captures principle properties of arguments:

1. *PageRank*. An argument’s relevance corresponds to the PageRank of its premises. This is the approach that we propose.
2. *Frequency*. An argument’s relevance corresponds to the frequency of its premises in the graph. This baseline captures popularity, as proposed in related work (see Section 2).
3. *Similarity*. An argument’s relevance corresponds to the similarity of its premises to its conclusion. We use the Jaccard similarity between all words in the premises and the conclusion. This basic content-oriented baseline quantifies the support of premises.
4. *Sentiment*. An argument’s relevance corresponds to the positivity of its premises. Here, we sum up the positive values of all premise words in SentiWordNet (Baccianella et al., 2010) and subtract all negatives. Also this baseline quantifies the support of premises.
5. *Most premises*. An argument’s relevance corresponds to its number of premises. This simple baseline captures the amount of support.
6. *Random*. The relevance is decided randomly. This baseline helps interpreting the results.

Experiment For all 32 conclusions of our benchmark rankings, we assessed the relevance of every associated argument with all six approaches—in case of 1.–4. once for each premise aggregation method. For all approaches, we then compared the resulting ranks with the respective benchmark ranks and computed the mean correlation over all conclusions in terms of Kendall’s τ . Kendall’s τ is most suitable here, as it is meant for ranks and as it applies even when all arguments are ranked equally (unlike, e.g., the Pearson coefficient).

Results Table 2 shows that the highest rank correlation is clearly achieved by our *PageRank* approach, namely, when using the *Sum* aggregation. While a Kendall’s τ of 0.28 is not very high, it can be interpreted as noncoincidental, and it is close to the low mean τ of all experts (0.36) resulting from subjectivity (see Section 5). *PageRank Sum* proves best in 15 of 32 cases. *Most premises*, which has the second highest τ (0.19), produced fewer worst rankings, but this is because it ranks all arguments equally for those 22 of the 32 conclusions where all have the same number of premises.

Matching the notion that popularity is not correlated with merit (see Section 2), *Frequency* hardly achieves anything. In fact, each frequency approach is outperformed by the PageRank approach with the respective aggregation method. The same holds for *Similarity*, which even seems to correlate rather negatively with relevance. An explanation may be that similarity rewards redundancy, which is why, e.g., all four similarity approaches falsely ranked the redundancy-free argument a_1 in Table 1 lowest. However, this requires further investigation, including an analysis of more sophisticated similarity measures. *Sentiment*, finally, performs comparably strong with $\tau > 0.1$ in three cases. In accordance with the second ranking in Table 1, this suggests that naming positive aspects (which support a conclusion) benefits relevance.

Regarding the four premise aggregation methods, we point out that their success might be partly affected by the scale of our data: 31,080 argument units is still tiny compared to the web argument graph we envision. In case of the first conclusion in Table 1, e.g., both *Minimum* and *Average* under-rate the relevance of a_1 , since *premise 1* fails to counter the low PageRank score of *premise 2* (resulting in $\tau = -0.82$). Summing up scores makes sense for these strongly connected premises, and it increases τ to 0.82. In contrast, *Maximum* assigns the same rank to all three arguments, which would be unlikely if web-scale data was given.

We conclude that a final judgment about our approach will require a web-scale analysis. Still, we saw first evidence for the impact of assessing argument relevance with PageRank. Considering that PageRank fully ignores an argument’s content and inference—unlike our human expert rankers—its observed dominance is quite intriguing.

7 Towards Argument Search Engines

From an application viewpoint, the long-term goal of our research on argument relevance is to enable web search engines to provide the most important arguments in response to queries on controversial topics. In this regard, the proposed PageRank approach serves to retrieve relevant candidate arguments. These arguments should then be further assessed, e.g., in terms of the soundness of their inference or other quality dimensions (Wachsmuth et al., 2017). At web scale, however, our approach poses several challenges of processing natural language text, most of which refer to the construction of a reliable argument graph.

The kind of construction process that we foresee starts with the language identification and content extraction of web pages, followed by linguistic preprocessing (sentence splitting, part-of-speech tagging, etc.). For major languages, the respective technologies are not perfect but reliable (Gottron, 2008). Then, argument mining is needed in order to segment and classify argument units as well as to compose arguments. While some mining approaches for web content exist, their robustness still needs improvement (Al-Khatib et al., 2016a). The most complex step is to identify the reuse of a conclusion as a premise in another argument. Ultimately, this implies that the units are semantically equivalent (or contradictory). Both textual entailment and paraphrasing help but are themselves un-

solved in general. At least, promising results with about 70% accuracy are reported for ground-truth arguments (Cabrio and Villata, 2012a).

Nevertheless, the goal of bringing argument relevance to practice is not at all a dream of the far future. The decisive observation is here that the size of the web allows preferring precision over recall. In particular, an initial high-precision, lower-recall argument graph may be obtained by focusing on “low-hanging fruits”. For instance, reliable arguments can be derived from those web sources that are directly cited in online debate portals, such as <http://www.debatepedia.org>. Generally, the mining process can be tailored to narrow domains and to well-structured text genres first. In order to limit the noise from mining errors, simple and unambiguous sentence-level arguments may be focused on and mined only if the respective approaches have a high confidence. Similarly, the recognition of equivalent argument units may be restricted to near-duplicates based on high-precision heuristics, such as ignoring capitalization, discourse connectives and other filler words, or similar.

From there on, the framework nature of the defined argument graph allows a stepwise refinement of the process, integrating new approaches to any process step as available. Research towards argument search engines can hence start now.

8 Conclusion

This paper proposes a model to integrate argument relevance in future web search, and it lays theoretical ground for research on argument relevance. In particular, we have defined how to construct an argument graph at web scale as well as how to adapt PageRank for arguments in order to objectively assess relevance given the graph. The results on our new, freely available Webis-ArgRank-17 benchmark dataset with a ground-truth argument graph of notable size suggest that PageRank outperforms both frequency-based and simple content-based relevance assessment approaches.

An evaluation at web scale is left to future work. Currently, we are working on approaches that robustly mine arguments from web pages, preferring precision over recall in order to obtain a more reliable argument graph. In general, several considerable challenges exist towards the argument search engines we envision, not only in terms of argument mining. We propose to face these challenges in order to shape the future of web search together.

References

- Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68. Association for Computational Linguistics.
- Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016a. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404. Association for Computational Linguistics.
- Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016b. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443.
- Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).
- Floris Bex, John Lawrence, Mark Snaith, and Chris Reed. 2013. Implementing the argument web. *Communications of the ACM*, 56(10):66–73.
- Yonatan Bilu, Daniel Hershcovich, and Noam Slonim. 2015. Automatic claim negation: Why, how and when. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 84–93. Association for Computational Linguistics.
- Stefano Bistarelli, Daniele Pirolandi, and Francesco Santini. 2011. Solving weighted argumentation frameworks with soft constraints. In *Recent Advances in Constraints: 14th Annual ERCIM International Workshop on Constraint Solving and Constraint Logic Programming*, pages 1–18. Springer Berlin Heidelberg.
- J. Anthony Blair. 2012. *Groundwork in the Theory of Argumentation*. Springer Netherlands.
- Filip Boltužić and Jan Šnajder. 2015. Identifying prominent arguments in online debates using semantic textual similarity. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 110–115. Association for Computational Linguistics.
- Liora Braunstain, Oren Kurland, David Carmel, Idan Szpektor, and Anna Shtok. 2016. Supporting human answers for advice-seeking questions in CQA sites. In *Proceedings of the 38th European Conference on IR Research*, pages 129–141.
- Elena Cabrio and Serena Villata. 2012a. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 208–212. Association for Computational Linguistics.
- Elena Cabrio and Serena Villata. 2012b. Natural language arguments: A combined approach. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 205–210.
- Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison-Wesley, USA, 1st edition.
- Pierpaolo Dondio. 2014. Toward a computational analysis of probabilistic argumentation frameworks. *Cybernetics and Systems*, 45(3):254–278.
- Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357.
- Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996. Association for Computational Linguistics.
- Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. 2010. Corroborating information from disagreeing views. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 131–140.
- Thomas Gottron. 2008. *Content Extraction — Identifying the Main Content in HTML documents*. Ph.D. thesis, Universität Mainz.
- Trudy Govier. 2010. *A Practical Study of Argument*. Wadsworth, Cengage Learning, Belmont, CA, 7th edition.
- Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingness of web arguments using bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics.
- Ralph H. Johnson and J. Anthony Blair. 2006. *Logical Self-defense*. International Debate Education Association.
- Jon M. Kleinberg. 1999. Hubs, authorities, and communities. *ACM Computing Surveys*, 31(4).
- Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of COLING*

- 2014, *the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500. Dublin City University and Association for Computational Linguistics.
- Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.
- Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.
- Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38. Association for Computational Linguistics.
- Jeff Pasternack and Dan Roth. 2010. Knowing what to believe (when you already know something). In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 877–885. Coling 2010 Organizing Committee.
- Andreas Peldszus and Manfred Stede. 2015. Towards detecting counter-considerations in text. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 104–109. Association for Computational Linguistics.
- Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552. Association for Computational Linguistics.
- Chris Reed and Glenn Rowe. 2004. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools*, 14:961–980.
- Paul Reisert, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. 2015. A computational approach for generating Toulmin model argumentation. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 45–55. Association for Computational Linguistics.
- Ruty Rinott, Lena Dankin, Carlos Alzate Perez, M. Mitesh Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence — An automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450. Association for Computational Linguistics.
- Mehdi Samadi, Partha Pratim Talukdar, Manuela M. Veloso, and Manuel Blum. 2016. ClaimEval: Integrated and flexible framework for claim evaluation using credibility of sources. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 222–228.
- Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510. Dublin City University and Association for Computational Linguistics.
- Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.
- Frans H. van Eemeren. 2015. *Reasonableness and Effectiveness in Argumentative Discourse: Fifty Contributions to the Development of Pragma-Dialectics*. Argumentation Library. Springer International Publishing.
- Nina Wacholder, Smaranda Muresan, Debanjan Ghosh, and Mark Aakhus. 2014. Annotating multiparty discourse: Challenges for agreement metrics. In *Proceedings of LAW VIII - The 8th Linguistic Annotation Workshop*, pages 120–128. Association for Computational Linguistics and Dublin City University.
- Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment flow — A general model of web review argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 601–611. Association for Computational Linguistics.
- Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.
- Marilyn Walker, Jean Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 812–817. European Language Resources Association (ELRA).
- Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.
- Douglas Walton. 2006. *Fundamentals of Critical Argumentation*. Cambridge University Press.
- Xiaoxin Yin, Jiawei Han, and Philip S. Yu. 2007. Truth discovery with multiple conflicting information providers on the web. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1048–1052.

Predicting Counselor Behaviors in Motivational Interviewing Encounters

Verónica Pérez-Rosas¹, Rada Mihalcea¹, Kenneth Resnicow²,
Satinder Singh¹, Lawrence An³, Kathy J. Goggin⁴ and Delwyn Catley⁵

¹Computer Science and Engineering, University of Michigan

²School of Public Health, University of Michigan

³Center for Health Communications Research, University of Michigan

^{4,5}Department of Pediatrics, Children's Mercy Kansas City, University of Missouri–Kansas City

^{1,2,3}{vrncapr, mihalcea, kresnic, baveja, lcan}@umich

^{4,5}{kjgoggin, dcatley}@cmh.edu

Abstract

As the number of people receiving psychotherapeutic treatment increases, the automatic evaluation of counseling practice arises as an important challenge in the clinical domain. In this paper, we address the automatic evaluation of counseling performance by analyzing counselors' language during their interaction with clients. In particular, we present a model towards the automation of Motivational Interviewing (MI) coding, which is the current gold standard to evaluate MI counseling. First, we build a dataset of hand labeled MI encounters; second, we use text-based methods to extract and analyze linguistic patterns associated with counselor behaviors; and third, we develop an automatic system to predict these behaviors. We introduce a new set of features based on semantic information and syntactic patterns, and show that they lead to accuracy figures of up to 90%, which represent a significant improvement with respect to features used in the past.

1 Introduction

Effective behavioral counseling is an essential element in combating public health issues such as mental health, substance abuse, and nutrition among others. A key component in training addiction counselors and other health care providers is providing detailed clinical feedback and evaluation. In clinical psychotherapy, this is done through behavioral coding, a labor intensive and time consuming process that requires highly trained practitioners who observe the counseling interactions via audio/video or reading transcripts and, then provide detailed evaluative feed-

back based on a set of predefined behaviors.

Recently, research efforts have been made towards implementing automatic means to assist this process and provide clinicians with tools to code and analyze counseling narratives (Atkins et al., 2014; Xiao et al., 2014; Klonek et al., 2015). Such tools can enable analyzes at larger scale by providing faster, cheaper, and more reliable methods for coding and data summarizing tasks.

Following this line of work, this paper presents a text-based approach for the automatic coding of counselor's verbal behaviors during counseling encounters. We focus our analysis on counseling conducted using Motivational Interviewing (MI), a well established evidence-based psychotherapy style, and the Motivational Interviewing Treatment Integrity (MITI) coding scheme.

2 Background on Motivational Interviewing

Miller and Rollnick define MI as a collaborative, goal-oriented style of psychotherapy with particular attention to the language of change (Miller and Rollnick, 2013). MI has been widely used as treatment method in clinical trials on psychotherapy research to address addictive behaviors such as alcohol, tobacco and drug use; promote healthier habits such as nutrition and fitness; and help clients with psychological problems such as depression and anxiety (Rollnick et al., 2008; Polak et al., 2010; Lundahl et al., 2010; Vader et al., 2010; Apodaca et al., 2014; Magill et al., 2014; Moyers and Martin, 2006; Moyers et al., 2009; Glynn and Moyers, 2010; Barnett et al., 2014). In addition, MI has been successfully applied in different practice settings including social work in behavioral health centers, education, and criminal justice (Wahab, 2005; McMurrin, 2009).

MI implementation requires effective counselor

training, supervision, and evaluation. Counselor's competence in MI delivery is measured by either focusing on counselor behaviors, client behaviors, or both (Jelsma et al., 2015).

The MITI coding system is currently the gold standard instrument for this task (Moyers et al., 2005). MITI focuses on counselors verbal behaviors and measures their MI proficiency by evaluating the use of reflective listening; questions; counselor strategies to engage clients such as seeking collaboration, affirming, and emphasizing autonomy; behaviors that indicate counselor deficiencies while delivering MI such as confronting and persuading without permission; and finally, neutral behaviors such as providing information and persuading with permission.

3 Related Work

Recently there have been a number of efforts on building computational tools that assist clinical psychotherapy on behavioral coding tasks.

(Can et al., 2012) proposed a linguistic based approach to automatically detect and code counselor reflections that is based on analyzing n-grams patterns, similarity features between counselor and client speech, and contextual meta-features, which aim to represent the dialog sequence between the client and counselor. A method based on labeled topic models is presented in (Atkins et al., 2012; Atkins et al., 2014), where authors focus on automatically identifying topics related to MI behaviors from the MISC scheme such as reflections, questions, support, and empathy. Unlike their work, we introduce and experiment with richer sets of features that represent more accurately the linguistic structure of counselor behaviors, including syntactic patterns and semantic information. Moreover, although we also focus on the recognition of the two most frequently encountered behaviors (reflections and questions), we also apply and evaluate our system on the other MI behaviors measures by the MITI coding scheme. Speech and linguistic based methods have also been proposed to evaluate overall MI quality. For instance, (Xiao et al., 2014) presents a study on the automatic evaluation of counselor empathy. The method is based on analyzing correlations between prosody patterns and empathy showed by the therapist during the counseling interactions.

Although most of the work on coding of MI

within session language has focused on modeling the counselor language, there is also work that investigates the client language. (Tanana et al., 2015) addresses the identification of counselor's statements discussing client's change talk. Their approach uses recursive neural networks to model sequences of counselor and client verbal exchanges. (Lord et al., 2015b) analyze the language style synchrony between therapist and client during MI encounters. They rely on the psycholinguistic categories from the Linguistic Inquiry and Word Count lexicon to measure the degree in which counselor language matches the client language.

Also related to our research is work on the social interaction domain. (Danescu-Niculescu-Mizil et al., 2012) studied power differences from language coordination in group discussions by measuring the similarity of word usage across different linguistic categories. Stylistic influence and symmetry have also been explored in social media interactions (Danescu-Niculescu-Mizil et al., 2011). More recently, (Althoff et al., 2016) explored these phenomena in the mental health domain by analyzing text-message-based counseling and observed that counselors who are more successful act with more control in the conversations and coordinate in a lower degree than their less successful counterparts.

In summary, research findings have shown that natural language processing approaches can be successfully applied to clinical narratives for the automatic annotation and analysis of therapists' and clients' behaviors. However, developed methods have not yet explored the use of linguistic features that incorporate semantic or syntactic information. In this paper we seek to explore new linguistic representations that can improve the identification of MITI counselor behaviors. Furthermore, we also experiment with features that measure participants linguistic accommodation during the counseling interaction.

4 MI Narratives Dataset

The data used in this study consists of 277 MI sessions conducted in several medical settings, including randomized control trials in clinical research for smoking cessation and medication adherence; MI training from a graduate-level MI course; wellness coaching phone calls; and brief medical encounters in dental practice and student

counseling. The full set comprises 97.8 hours of audio with an average session length of 20.8 minutes with a standard deviation of 11.5 minutes.

4.1 Transcription

Before transcribing, all the counseling recordings were preprocessed to remove any personal identifiers. This includes manually trimming the audio to remove introductions and replacing references to participant's name and location with silences.

Sessions were transcribed via Mechanical Turk (Marge et al., 2010) using the following guidelines: 1) transcribe speech turn by turn, 2) clearly identify the speaker (either client or counselor), 3) include speech disfluences, such as false starts, repetitions of whole words or parts of words, prolongations of sounds, fillers, long pauses. Transcriptions were manually verified at random points to avoid spam and ensure their quality. The final transcript set contains 22,719 utterances.

4.2 MI Coding

MITI coding was conducted by a team of three counselors who have extensive experience with MI.¹ Prior to the annotation phase, annotators participated in a coding calibration step where they discussed the criteria for sentence parsing, the correct assignment of behavior codes, and conducted team coding in a set of sample sessions.

As suggested in the MI literature, we evaluated the coding reliability on a sample of ten double-coded sessions, which were coded by our staff and by MITI developers (Moyers et al., 2005).

We measured the inter-annotator agreement at both session and utterance level. For the session level, we measured the Intraclass Correlation Coefficient (ICC), which indicates how much of the total variation in MITI scores is due to differences among annotators (Dunn et al., 2015). The utterance level agreement was measured using the Kappa score (Lord et al., 2015a).

The ICC values reported in Table 1 show noticeable high agreement for the Question and Reflection codes with scores ranging between 0.89 to 0.97, which are considered excellent agreement in the MI literature (Jelsma et al., 2015). The remaining codes show lower agreement values due to low frequency counts in the sample. This was particularly the case for the Giving Information, Affirm

¹Annotators were trained in the use of MITI 4.1 by expert trainers from the Motivational Interviewing Network of Trainers

and Emphasizing Autonomy codes, for which we were unable to obtain ICC scores (NA). Confront, and Persuading without Permission codes are not reported as they did not appear in our sample. The main reason for this is that the dataset was derived from sessions conducted by experienced counselors who avoided such codes as they indicate bad MI practice.

Overall, the ICC scores suggest that the annotators do not show significant variations at session level coding, i.e., the total frequency counts of each code per session did not differ significantly between coders. Furthermore, the Kappa scores suggest that annotators have fair to good pairwise agreement at utterance level coding.

Since the inter-reliability analysis showed reasonable agreement among the coding team members, we moved forward to the annotation phase. The 277 sessions are randomly distributed among the three members of the coding team. Annotations are conducted using the session audio recording along with its transcript using Nvivo,² a quantitative analysis suite for behavioral coding that allows selecting free text and assigning it to a given category. Table 2 presents an excerpt of a session transcript. As observed, a talk-turn can comprise multiple utterances.

The team annotated approximately 20 sessions per week. The entire annotation process took nearly three months. After the annotation phase, the annotated transcripts were processed to extract the verbal content of each MITI annotation; non-coded utterances were also extracted and labeled as neutral speech. In the coded set 33% (5262) are Questions, 17% (2690) are Simple Reflections, 18% (2876) are Complex Reflections, and 32% (5058) are other MITI codes: Seeking Collaboration (614), Emphasizing Autonomy (141), Affirm (499), Confront (141), Persuading without Permission (598), Giving information (1017), and Persuading with Permission (2100).

5 Linguistic Features for MI behaviors

In order to explore linguistic patterns related to counselor behaviors, we analyze their definitions and usage. For instance, the use of reflective statements helps counselors understand client's statements through hypothesis testing (Miller and Rollnick, 2013); questions help counselor elicit information and engage the clients in the conversation;

²<http://www.qsrinternational.com/what-is-nvivo>

Transcript		Code
T	<i>So, before we go further, you know, there's two different aspects of, you know, of weight. So there's the food aspect and the exercise aspect. Is there something that you'd particularly like to focus on today?</i>	GI,QUEST
C	Well I think my-my biggest concern is the food issue, and h-how to eat better. So, I think I'd like to start there.	
T	Okay. <i>Because you mentioned that you've been active before in sports</i>	S-REFL
C	Yeah, and, you know, with the nice weather coming, I'd like to get outside and do things, so I'm sure that will come, you know, soon.	
T	Right.	
C	I just, you know, it's so hard to-to change my eating habits.	
T	<i>So, it sounds like, you know, you may even feel, sort of, more confident that you'll be more active physically. And then that's why you'd like to focus on the food part. Because if you know that that's coming up and you're sure that you will be able to do that, then the food part would really help.</i>	C-REFL
C	Yeah, exactly.	

Table 2: Transcript excerpt from a MI session between therapist (T) and client (C). MI codes include: Complex Reflection (C-Refl), Simple Reflection (S-Refl), Question (Quest), Giving information (GI). Coded utterances are shown in italics.

Behavior	Inter-reliability	
	ICC	Kappa
Question	0.97	0.64
Complex reflection	0.97	0.49
Simple reflection	0.89	0.34
Seeking collaboration	0.03	0.42
Giving Information	NA	0.28
Affirm	NA	0.47
Emphasizing autonomy	NA	0.31

Table 1: Inter-annotator agreement for the MI dataset in a random sample of 10 sessions

and so on. Considering these guidelines, we derive the following features that aim to capture the linguistic differences among these behaviors.

N-grams: These features represent the language used by the counselor and include all the unique words and word-pairs present in counselor speech. We extract a vector containing the frequencies of each word and word pair present in each utterance.

Semantic information: These features attempt to bring semantic information into the analysis of counselor language by identifying words as belonging to certain semantic categories that are potential markers of counseling style. For instance, semantic categories related to reflective language include tentative language e.g., maybe, perhaps, looks, as well as anxiety words e.g., afraid, tense, worried. We use two groups of semantic features. The first consists of features derived from the LIWC lexicon (Tausczik and Pennebaker, 2010), a psycholinguistic resource that contains 70 semantic categories representing psy-

chological cues to human thought processes, emotional states, intentions, and motivations. The second is a self-acquired reflection lexicon consisting of 146 words frequently present during reflective statements. These features are represented as the total frequency counts of all the words in a word category that are present in the annotation.

Similarity: Since reflective listening includes repetition and rephrasing, we can expect to observe linguistic similarity between client and counselor speech. Thus, we measure the degree to which the counselor matches the client language by using Linguistic Style Matching (LSM) (Gonzales et al., 2009), a technique that allows to quantify the extent to which one person uses comparable types of words to another person. We measure LSM at a turn-by-turn level using the LIWC word categories, e.g., positive words, pronouns, negations, quantifiers. In order to capture information from return statements, we combine client speech from the previous and current turn along with the counselor utterance. The features are represented by a score ranging between 0 and 1 indicating the degree to which the counselor and client use the same type of words.

Syntactic features: These features aim to represent the syntactic structure of the clinician statements. We use these features to encode information about the word order in the sentence. We expect syntactic patterns with high occurrence will likely capture reflection starters commonly used

Features	Acc.	P	R	F
REFL vs. ALL				
Baseline	75.00%	0.00	0.00	0.00
N-grams	82.47%	0.69	0.66	0.67
Semantic	77.37%	0.68	0.34	0.45
Similarity	60.95%	0.58	0.34	0.42
Syntax	78.65%	0.72	0.67	0.62
All features	82.62%	0.69	0.66	0.67
REFL vs. OTHER				
Baseline	64.00%	0.00	0.00	0.00
N-grams	83.51%	0.78	0.80	0.79
Semantic	71.57%	0.68	0.54	0.58
Similarity	63.00%	0.58	0.27	0.37
Syntax	86.80%	0.82	0.86	0.84
All features	81.27%	0.84	0.83	0.84

Table 3: Classification results for counselor reflections (REFL), other MITI codes (OTHER), other MITI codes + transition (unannotated) utterances (ALL)

by the counselor such as “it sounds like ...”. First, we use the Stanford parser to generate the Context Free Grammar parse trees of counselor utterances and extract all production rules present in the trees. Second, we derive features for each lexicalized and unlexicalized production rule augmented with its grandparent node; this means we also include chunk tags such as noun phrases, adverb phrases, prepositional phrases, and so on. Third, each feature is calculated by counting how many times a production rule or production-rule-sequence occurs in the utterance.

6 Experiments and Results

After the feature extraction, we explore whether these features can be used as predictors of counselor behaviors. We first focus on the prediction of reflections and questions, as they represent the most frequent behaviors in counseling narratives; we then experiment with the use of these features for the prediction of the other behavior codes.

6.1 Predicting Counselor Reflections

We conduct learning experiments where we explore the use of n-grams, syntactic, semantic, and similarity features to build reflection classifiers at three levels of detail.

First, we attempt to mimic the process human coders follow while MITI annotating a session, i.e., the coder goes through each counselor utterance and chooses the most appropriate code according to the MITI guidelines. Hence, we focus on the identification of Reflection utterances regardless of being complex or simple. The learning

task aims to classify a counselor utterance either as a Reflection, or a Not-reflection, i.e., any other counselor utterance. Second, given that a large portion of the verbal exchanges between the counselor and the client consists of transition or facilitative statements (e.g., yeah, right), we decided to remove this content from the analysis thus focusing on the task of discriminating between Reflection and any other MITI code. Third, we aim to discriminate between Simple and Complex reflection. In MI, counselors use both types of reflections to understand the client’s perspective, feelings, and values. However, in general, complex reflections are preferable over simple reflections as they show counselor’s deeper understanding of the issues being discussed. In a real setting, distinguishing between these two behaviors and understanding their linguistic differences is important in order to provide the counselor with feedback on the nature of their reflective statements.

During our experiments we employ the Support Vector Machines (SVM) (Cortes and Vapnik, 1995) classifier as the main classifier. We use the version implemented in the LibLinear library with the default parameters. We build several classification models using each of the different sets of linguistic features. We evaluate the ability of such models to predict the target behavior using a five-fold cross-validation. As reference value, we use a majority class baseline, which is the percentage of instances correctly classified when selecting by default the most frequent category in the training data.

Table 3 summarizes the classification performance for each set of features in the detection of reflections. During our experiments we used F-score as the main evaluation metric. This metric considers both the proportion of reflections identified from the training set (recall) and the proportion of reflections correctly identified as such (precision). From this table, we can observe that the syntactic model accurately captures differences between reflective and non-reflective content. This difference is even more noticeable when discriminating between syntactic structures associated to reflective statements versus syntactic structures associated to other MITI codes (REFL vs OTHER column).

Table 4 presents the classification performance for the simple (S-Refl) and complex reflections (C-Refl). F-scores among the classification mod-

Features	Acc.	S-REFL			C-REFL		
		P	R	F	P	R	F
Baseline	52.00%	0.00	0.00	0.00	0.52	1.00	0.68
N-grams	63.24%	0.61	0.65	0.64	0.66	0.62	0.63
Semantic	67.21%	0.63	0.65	0.64	0.67	0.70	0.69
Similarity	63.22%	0.62	0.59	0.58	0.67	0.58	0.63
Syntax	65.06%	0.63	0.66	0.64	0.67	0.70	0.657
All features	62.52%	0.60	0.62	0.61	0.64	0.62	0.63

Table 4: Classification results for Simple Reflections (S-Refl) and Complex Reflections (C-Refl)

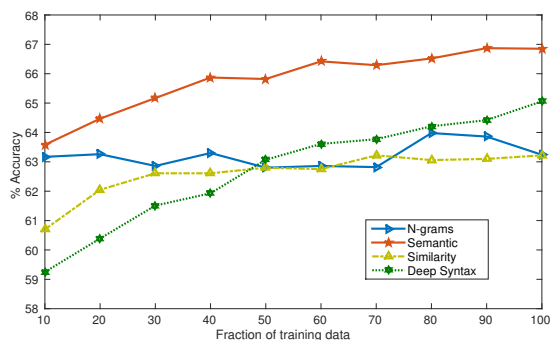


Figure 1: Learning curve for Simple (S-Refl) and Complex Reflection (C-Refl) detection using different amounts of training data and four feature sets.

els do not show noticeable improvement gain for the syntactic model. This suggest that the syntactic features might not have enough discriminative power to accurately differentiate between Complex and Simple reflections as the differences among these behavior codes are mostly semantic. This could be also attributed to the similarity of syntactic constructions for Simple and Complex reflections, i.e., common starters such as *it sounds like, it looks like*, and other similar syntactic constructions. Note that this task is also challenging for humans, as reported pairwise inter-annotator agreements for Simple and Complex reflections in our sample ranges between fair to good levels (see Kappa values in Table 1).

Finally, we investigate whether larger amounts of training data can be helpful to discriminate between Simple and Complex reflections, in particular with the use of syntax-based features. We plot the learning curves of the different sets of features using incremental amounts of data as shown in Figure 1. The learning trend suggests the classification performance while distinguishing between Complex and Simple Reflection improves when increasing the number of training examples. Notably, the syntactic features curve shows consistent growth, suggesting that larger quantities of train-

Target Behavior Change	Sessions	REF	OTHER
Medication adherence	93	2977	4031
Smoking Cessation	95	2290	3745
Dietary Changes	72	2045	2669

Table 5: Class distribution for three target behavior changes

ing data might improve the classification performance for this task.

6.2 The Role of Behavior Change Target

During MI encounters, counselors follow specific strategies to guide the client towards behavior change. For instance, reflective listening strategies include generic starters to reformulate, rephrase, or intensify client’s statements, which are used regardless of the behavior change target. Considering that MI has proven to be effective on addressing a wide range of application domains, this might suggest a certain degree of domain independence, which can be of importance for the development of natural language processing strategies for the automatic coding of MI sessions.

Aiming to explore the role played by the health issue being discussed during the counseling encounter, we conduct an additional set of experiments on three target behavior changes present in our dataset, namely medication adherence, smoking cessation, and dietary changes. The class distribution for each set is shown in Table 5. We exclude 16 sessions as they correspond to miscellaneous change goals.

Using this data, we build reflection classifiers using the linguistic features described before. Results are shown in Table 6. From this table, we can derive interesting observations. First, following a similar trend as in our previous experiments, syntactic features offer improved performance over the n-grams, similarity, and semantic feature sets. Second, we observe more steady improvement when using a combination of different feature sets, as compared to our previous experi-

Target Behavior Change	Baseline	N-grams	Semantic	Similarity	Syntax	All Features
Medication adherence	57.52%	83.56%	61.48%	57.52%	85.31%	88.78%
Smoking cessation	62.05%	82.41%	66.16%	62.96%	83.41%	85.34%
Dietary changes	56.51%	82.75%	66.50%	59.56%	82.42%	85.41%

Table 6: Classification performance for reflection detection (REFL VS. ALL) on sessions aiming at three behavior changes

Category	Medication Adherence	Smoking Cessation	Dietary Changes
Noun	health, family, routine, life	children, control, past, role, parent	pre-diabetes, people, husband, future
Verb	live, imagine, see, eat, help	see, affect, feel, want, talk	care, affect, concern, leave
Adjective	difficult, responsible, important	concern, hard, helpful	scary, busy, difficult, willing

Table 7: Counselor word usage across different health issues

ments. Still, the performance for both sets of experiments is comparable, thus suggesting that the task is not heavily affected by the health issue being discussed. This further suggests that training data on the same behavior target is desirable but not essential.

Since we did not observe noticeable differences in language constructions used by counselors across different health issues, we decided to analyze whether counselors differ in their word choices. We thus looked at the top syntactic features generated for each classification model and their corresponding terminal nodes and part of speech tags. Table 7 shows sample words for nouns, verbs, and adjectives used by counselors while formulating reflections for three target behavior changes. From this table we notice that counselor word usage does vary with the health issue being addressed. For instance, when discussing smoking cessation, counselor emphasize verbs and nouns that evoke clients’ desire to change (affect, want, feel) and discuss client values that are related to how they are perceived by others (role, parent, control).

6.3 Prediction of Counselor Questions

Our next set of experiments aims to predict counselor questioning statements. As before, we build different prediction models using the developed feature sets and attempt to discriminate between 1) Questions and any other counselor utterance (QUEST vs ALL), and 2) Questions and other MITI codes (QUEST vs OTHER). Classification performances for these models are shown in Table 8. The best performing feature set is the syntactic followed by the n-grams model.

Note that in addition to the experiments reported in this table, we attempted to combine different features sets. However, we did not observe

Features	Acc.	P	R	F
QUEST vs. ALL				
Baseline	76.83%	0.00	0.00	0.00
N-grams	87.81%	0.91	0.92	0.76
Semantic	79.19%	0.69	0.37	0.48
Similarity	62.33%	0.36	0.31	0.36
Syntax	90.59%	0.82	0.81	0.81
All features	81.87%	0.76	0.74	0.75
QUEST vs. OTHER				
Baseline	66.87%	0.00	0.00	0.00
N-grams	88.84%	0.86	0.85	0.85
Semantic	75.28%	0.70	0.64	0.67
Similarity	57.33%	0.38	0.49	0.42
Syntax	90.48%	0.92	0.92	0.87
All features	86.57%	0.82	0.82	0.82
GRU	92.8%	0.89	0.92	0.90

Table 8: Classification results for counselor questions (QUEST), other MITI codes + transition (unannotated) utterances (ALL), and other MITI codes (OTHER).

substantial improvement over our best performing model consisting of syntactic features.

6.4 Prediction of Other MI Codes

Aiming to identify potential predictors for the remaining MI codes, we conduct a set of experiments where we use our linguistic feature sets to build multiclass classifiers. Table 9 shows the precision and recall figures obtained for the different classification models. Note that the results using semantic and similarity features are not reported, as the resulting classifiers showed very low recall values. From these results we observe that both syntactic features and n-grams aid the identification of other counselor behavior codes, particularly Giving Information, Affirm, and Seeking Collaboration. However, the prediction accuracy of the syntactic models is slightly lower than the n-grams models. We believe that the more verbose nature of these codes, in contrast to reflections, makes it difficult to benefit from syntactic patterns.

Features	GI		AF		SEEK		AUTO		PWP		PWOP		CON	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R
N-grams	0.56	0.47	0.42	0.37	0.55	0.49	0.29	0.26	0.23	0.15	0.29	0.22	0.10	0.07
Syntax	0.52	0.47	0.41	0.35	0.54	0.50	0.26	0.19	0.18	0.14	0.28	0.22	0.10	0.04
GRU	0.48	0.79	0.28	0.75	0.41	0.80	0.24	0.30	0.09	0.47	0.21	0.38	0.05	0.17

Table 9: Classification results for seven MI behaviors: Giving Information (GI), Affirm (AF), Seeking Collaboration (SEEK), Emphasizing Autonomy (AUTO), Persuading with Permission (PWP), Persuading without Permission (PWOP), Confront (CON)

	QUEST	REFL	S-REFL	C-REFL
1	*. ^ S → ?	*VBZ ^ VP → sounds	ROOT ^ ROOT → S	VP ^ S → TO_VP
2	*. ^ SBARQ → ?	*IN ^ SBAR → since	*VBD ^ VP → mentioned	NP ^ PP → PRP\$ _NN
3	ROOT ^ ROOT → SBARQ	*IN ^ S → so	ROOT ^ ROOT → FRAG	*VBZ ^ VP → sounds
4	NP ^ SQ → PRP	S ^ ROOT → IN _NP	ROOT ^ ROOT → NP	*VB ^ VP → be
5	*.SQ → ?	VP ^ S → VBZ _SBAR	VP ^ S → VBD _SBAR	*TO ^ VP → to
6	ROOT ^ ROOT → SQ	*PRP ^ NP → it	*RB ^ ADVP → so	*RB ^ ADVP → really
7	*WP ^ WHNP → what	*RB ^ ADVP → really	VP ^ S → VBD _NP	*PRP ^ NP → it
8	*IN ^ PP → about	S ^ ROOT → CC _ADVP	*NN ^ NP → ok	*IN ^ SBAR → like
9	*DT ^ NP → any	ADJP ^ VP → RBR _JJ	*UH ^ INTJ → okay	*IN ^ PP → like
10	*. ^ FRAG → ?	VP ^ S → VBP _PRT	VP ^ S → VBD _PP	*DT ^ NP → this

Table 10: Most discriminative syntactic features for Questions (QUEST), Reflections (REFL), Simple reflection (S-REFL) and Complex reflection (C-REFL).

Also, note that Emphasizing Autonomy, Persuading With And Without Permission, and the Confront codes lead to low precision and recall values, which can be partially attributed to having a smaller number of training examples as compared to the other codes.

7 Discussion

Our experimental results support the use of automatic means to predict MITI counselor behaviors. Unsurprisingly, better results are obtained for the more frequent behaviors such as reflections and questions. Unlike previous studies that focused on the identification of reflective content in psychotherapy narratives (Atkins et al., 2014; Xiao et al., 2014), we build prediction models that predict all the MITI behavior codes. We also introduce and leverage new features consisting of semantic and syntactic patterns; our experimental results suggest the effectiveness of these new features.

To gain further insight into the syntactic patterns, we extract the most predictive features for each classification model. Table 10 presents a summary of the top ten production rules associated to Question (Quest), Reflection (Refl), Simple Reflection (S-Refl), and Complex Reflection (C-Refl). As expected, question production rules include the question mark as a clear indicator of questions. However, we also observe clause tags and phrase tags that capture more complex questioning structures such as direct questions in-

troduced by a wh-word or a wh-phrase *SBARQ*, inverted yes/no questions *SQ*, question personal pronoun *WP* (wh-pronoun, personal), and noun phrases *WHNP*.

Similarly, the most predictive rules for Reflections (REFL column) include adverbs (*RB*, *RBR*), adjectives (*JJ*), present tense verbs (*VBZ*), personal pronouns (*PRP*); as well as adverb and adjective phrases (*ADVP*, *ADJP*). We observe that the syntactic similarity of reflective statements is well represented by the syntactic model as they include verbal structures that are frequently used by the counselor to formulate reflective statements; for instance, generic reflection starters such as “So, it sounds like ...” (see rules 1, 3, 5, and 6 in column REFL), and word categories, such as adjectives, conjunctions and comparative adverbs (see rules 8, 9, and 10 in column REFL). Moreover, we observe an interesting difference in the verb tense usage for Simple and Complex Reflection detection: production rules for Simple Reflection include present tense while production rules for Complex Reflection include past tense.

Overall, our experimental results show the potential of applying linguistic methods in the prediction of counselor behaviors, and in particular those that incorporate syntactic information into the analysis.

8 Conclusions

In this paper, we presented a classification model towards the automation of MI coding using the MITI coding system.

We made two important contributions. First, we introduced a novel large psychotherapy dataset derived from MI interventions, consisting of 277 MI sessions with a total of 22,719 utterances. The dataset was manually transcribed and annotated with ten counselor verbal behaviors. Second, using several features, we applied the classification model to the recognition of MI counseling behaviors, with an emphasis on the two most frequently encountered behaviors: reflections and questions. We showed how a richer feature set, and in particular a set consisting of semantic and syntactic patterns, can lead to accuracy figures of up to 90%, which represents a significant improvement with respect to the bag-of-word features used in the past.

We also presented several analyses, including an exploration of the role of the behavior change target in the prediction of reflections; and an analysis of the most discriminative features in the syntactic model. Although this study focused on the MITI as the coding system and MI as the counseling approach, we believe that the proposed methods could apply to other measures of MI skill fidelity such as Behavior Change Counselor Index (BECCI) (Lane et al., 2005), Independent Tape Rating Scale (ITRS) (Martino et al., 2009), Stimulated Client Interview Rating Scale (SCIRS) (Arthur, 1999), and the One Pass coding system (McMaster and Resnicow, 2015).

Acknowledgments

This material is based in part upon work supported by the University of Michigan under the M-Cube program, by the National Science Foundation (grant #1344257), the John Templeton Foundation (grant #48503), and the Michigan Institute for Data Science. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the University of Michigan, the National Science Foundation, the John Templeton Foundation, or the Michigan Institute for Data Science. We gratefully acknowledge the help of the three counselors who helped with the data annotations.

References

- Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale Analysis of Counseling Conversations: An Application of Natural Language Processing to Mental Health. *Transactions of the Association for Computational Linguistics*.
- Timothy R. Apodaca, Brian Borsari, Kristina M. Jackson, Molly Magill, Richard Longabaugh, Nadine R. Mastroleo, and Nancy P. Barnett. 2014. Sustain talk predicts poorer outcomes among mandated college student drinkers receiving a brief motivational intervention. *Psychology of Addictive Behaviors*, 28(3):631.
- David Arthur. 1999. Assessing nursing students' basic communication and interviewing skills: the development and testing of a rating scale. *Journal of Advanced Nursing*, 29(3):658–665.
- David C. Atkins, Timothy N. Rubin, Mark Steyvers, Michelle A. Doeden, Brian R. Baucom, and Andrew Christensen. 2012. Topic models: A novel method for modeling couple and family text data. *Journal of family psychology*, 26(5):816.
- David C. Atkins, Mark Steyvers, Zac E. Imel, and Padhraic Smyth. 2014. Scaling up the evaluation of psychotherapy: evaluating motivational interviewing fidelity via statistical text classification. *Implementation Science*, 9(1):49.
- Elizabeth Barnett, Theresa B. Moyers, Steve Sussman, Caitlin Smith, Louise A. Rohrbach, Ping Sun, and Donna Spruijt-Metz. 2014. From counselor skill to decreased marijuana use: Does change talk matter? *Journal of substance abuse treatment*, 46(4):498–505.
- Dogan Can, Panayiotis G. Georgiou, David C. Atkins, and Shrikanth S. Narayanan. 2012. A case study: Detecting counselor reflections in psychotherapy for addictions using linguistic features. In *INTERSPEECH*, pages 2254–2257. ISCA.
- Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.
- Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words! Linguistic style accommodation in social media. In *Proceedings of WWW*, pages 745–754.
- Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*, pages 699–708.
- Chris Dunn, Doyanne Darnell, Sheng Kung Michael Yi, Mark Steyvers, Kristin Bumgardner, Sarah Peregrine Lord, Zac Imel, and David C. Atkins. 2015. Should we trust our judgments about the proficiency of motivational interviewing counselors? a glimpse at the impact of low inter-rater reliability. *Motivational Interviewing: Training, Research, Implementation, Practice*, 1(3):38–41.
- Lisa H. Glynn and Theresa B. Moyers. 2010. Chasing change talk: The clinician's role in evoking client language about change. *Journal of substance abuse treatment*, 39(1):65–70.
- Amy L. Gonzales, Jeffrey T. Hancock, and James W. Pennebaker. 2009. Language style matching as a predictor of social dynamics in small groups. *Communication Research*.

- Judith G.M. Jelsma, Vera-Christina Mertens, Lisa Forsberg, and Lars Forsberg. 2015. How to measure motivational interviewing fidelity in randomized controlled trials: Practical recommendations. *Contemporary clinical trials*, 43:93–99.
- Florian E. Klonek, Vicenç Quera, and Simone Kauffeld. 2015. Coding interactions in motivational interviewing with computer-software: What are the advantages for process researchers? *Computers in Human Behavior*, 44:284–292.
- Claire Lane, Michelle Huws-Thomas, Kerenza Hood, Stephen Rollnick, Karen Edwards, and Michael Rollins. 2005. Measuring adaptations of motivational interviewing: the development and validation of the behavior change counseling index (becci). *Patient education and counseling*, 56(2):166–173.
- Sarah Peregrine Lord, Doğan Can, Michael Yi, Rebeca Marin, Christopher W. Dunn, Zac E. Imel, Panayiotis Georgiou, Shrikanth Narayanan, Mark Steyvers, and David C. Atkins. 2015a. Advancing methods for reliably assessing motivational interviewing fidelity using the motivational interviewing skills code. *Journal of substance abuse treatment*, 49:50–57.
- Sarah Peregrine Lord, Elisa Sheng, Zac E. Imel, John Baer, and David C. Atkins. 2015b. More than reflections: Empathy in motivational interviewing includes language style synchrony between therapist and client. *Behavior therapy*, 46(3):296–303.
- Brad W. Lundahl, Chelsea Kunz, Cynthia Brownell, Derrick Tollefson, and Brian L. Burke. 2010. A meta-analysis of motivational interviewing: Twenty-five years of empirical studies. *Research on Social Work Practice*.
- Molly Magill, Jacques Gaume, Timothy R. Apodaca, Justin Walthers, Nadine R. Mastroleo, Brian Borsari, and Richard Longabaugh. 2014. The technical hypothesis of motivational interviewing: A meta-analysis of mis key causal model. *Journal of consulting and clinical psychology*, 82(6):973.
- Matthew Marge, Satanjeev Banerjee, Alexander Rudnicky, et al. 2010. Using the amazon mechanical turk for transcription of spoken language. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5270–5273. IEEE.
- Steve Martino, Samuel A. Ball, Charla Nich, Tami L. Frankforter, and Kathleen M. Carroll. 2009. Informal discussions in substance abuse treatment sessions. *Journal of substance abuse treatment*, 36(4):366–375.
- Fiona McMaster and Ken Resnicow. 2015. Validation of the one pass measure for motivational interviewing competence. *Patient education and counseling*, 98(4):499–505.
- Mary McMurran. 2009. Motivational interviewing with offenders: A systematic review. *Legal and Criminological Psychology*, 14(1):83–100.
- William R. Miller and Stephen Rollnick. 2013. *Motivational interviewing: Helping people change, Third edition*. The Guilford Press.
- Theresa B. Moyers and Tim Martin. 2006. Therapist influence on client language during motivational interviewing sessions. *Journal of substance abuse treatment*, 30(3):245–251.
- Theresa B. Moyers, Tim Martin, Jennifer K. Manuel, Stacey M.L. Hendrickson, and William R. Miller. 2005. Assessing competence in the use of motivational interviewing. *Journal of substance abuse treatment*, 28(1):19–26.
- Theresa B. Moyers, Tim Martin, Jon M. Houck, Paulette J. Christopher, and J. Scott Tonigan. 2009. From in-session behaviors to drinking outcomes: a causal chain for motivational interviewing. *Journal of consulting and clinical psychology*, 77(6):1113.
- Kathryn I. Pollak, Stewart C. Alexander, Cynthia J. Coffman, James A. Tulskey, Pauline Lyna, Rowena J. Dolor, Iguehi E. James, Rebecca J. Namenek Brouwer, Justin R.E. Manusov, and Truls Østbye. 2010. Physician communication techniques and weight loss in adults: Project chat. *American journal of preventive medicine*, 39(4):321–328.
- Stephen Rollnick, William R. Miller, Christopher C. Butler, and Mark S. Aloia. 2008. Motivational interviewing in health care: helping patients change behavior. *COPD: Journal of Chronic Obstructive Pulmonary Disease*, 5(3):203–203.
- Michael Tanana, Kevin Hallgren, Zac Imel, David Atkins, Padhraic Smyth, and Vivek Srikumar. 2015. Recursive neural networks for coding therapist and patient behavior in motivational interviewing. *NAACL HLT 2015*, page 71.
- Yla R Tausczik and James W. Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.
- Amanda M. Vader, Scott T. Walters, Gangamma Chenenda Prabhu, Jon M. Houck, and Craig A. Field. 2010. The language of motivational interviewing and feedback: counselor language, client language, and client drinking outcomes. *Psychology of Addictive Behaviors*, 24(2):190.
- Stephanie Wahab. 2005. Motivational interviewing and social work practice. *Journal of Social Work*, 5(1):45–60.
- Bo Xiao, Daniel Bone, Maarten Van Segbroeck, Zac E. Imel, David C. Atkins, Panayiotis G. Georgiou, and Shrikanth S. Narayanan. 2014. Modeling therapist empathy through prosody in drug addiction counseling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Authorship Attribution Using Text Distortion

Efstathios Stamatatos

University of the Aegean
83200, Karlovassi, Samos, Greece
stamatatos@aegean.gr

Abstract

Authorship attribution is associated with important applications in forensics and humanities research. A crucial point in this field is to quantify the personal style of writing, ideally in a way that is not affected by changes in topic or genre. In this paper, we present a novel method that enhances authorship attribution effectiveness by introducing a text distortion step before extracting stylometric measures. The proposed method attempts to mask topic-specific information that is not related to the personal style of authors. Based on experiments on two main tasks in authorship attribution, closed-set attribution and authorship verification, we demonstrate that the proposed approach can enhance existing methods especially under cross-topic conditions, where the training and test corpora do not match in topic.

1 Introduction

Authorship attribution is the task of determining the author of a disputed text given a set of candidate authors and samples of their writing (Juola, 2008; Stamatatos, 2009). This task has gained increasing popularity since it is associated with important forensic applications, e.g., identifying the authors of anonymous messages in extremist forums, verifying the author of threatening email messages, etc. (Abbasi and Chen, 2005; Lambers and Veenman, 2009; Coulthard, 2013), as well as humanities and historical research, e.g., unmasking the authors of novels published anonymously or under aliases, verifying the authenticity of literary works by specific authors, etc. (Koppel and Seidman, 2013; Juola, 2013; Stover et al., 2016).

The majority of published works in authorship

attribution focus on *closed-set attribution* where it is assumed that the author of the text under investigation is necessarily a member of a given well-defined set of candidate authors (Stamatatos et al., 2000; Gamon, 2004; Escalante et al., 2011; Schwartz et al., 2013; Savoy, 2013; Seroussi et al., 2014). This setting fits many forensic applications where usually specific individuals have access to certain resources, have knowledge of certain issues, etc. (Coulthard, 2013) A more general framework is *open-set attribution* (Koppel et al., 2011). A special case of the latter is *authorship verification* where the set of candidate authors is singleton (Stamatatos et al., 2000; van Halteren, 2004; Koppel et al., 2007; Jankowska et al., 2014; Koppel and Winter, 2014). This is essentially a one-class classification problem since the negative class (i.e., all texts by all other authors) is huge and extremely heterogeneous. Recently, the verification setting became popular in research community mainly due to the corresponding PAN shared tasks (Stamatatos et al., 2014; Stamatatos et al., 2015).

In authorship attribution it is not always realistic to assume that the texts of known authorship and the texts under investigation belong in the same genre and are in the same thematic area. In most applications, there are certain restrictions that do not allow the construction of a representative training corpus. Unlike other text categorization tasks, a recent trend in authorship attribution research is to build cross-genre and cross-topic models, meaning that the training and test corpora do not share the same properties (Kestemont et al., 2012; Stamatatos, 2013; Sapkota et al., 2014; Stamatatos et al., 2015).

One crucial issue in any authorship attribution approach is to quantify the personal style of authors, a line of research also called stylometry (Stamatatos, 2009). Ideally, stylometric fea-

tures should not be affected by shifts in topic or genre variations and they should only depend on personal style of the authors. However, it is not yet clear how the topic/genre factor can be separated from the personal writing style. Function words (i.e., prepositions, articles, etc.) and lexical richness features are not immune to topic shifts (Mikros and Argiri, 2007). In addition, character n -grams, the most effective type of features in authorship attribution as demonstrated in multiple studies (Grieve, 2007; Stamatatos, 2007; Luyckx and Daelemans, 2008; Escalante et al., 2011) including cross-topic conditions (Stamatatos, 2013; Sapkota et al., 2015), unavoidably capture information related to theme and genre of texts. Features of higher level of analysis, including measures related to syntactic or semantic analysis of texts, are too noisy and less effective, and can be used as complement to other more powerful low-level features (van Halteren, 2004; Argamon et al., 2007; Hedegaard and Simonsen, 2011).

In this paper, we propose a novel method that is based on text distortion to compress topic-related information. The main idea of our approach is to transform input texts to an appropriate form where the textual structure, related to personal style of authors, is maintained while the occurrences of the least frequent words, corresponding to thematic information, are masked. We show that this distorted view of text when combined with existing authorship attribution methods can significantly improve their effectiveness under cross-topic conditions in both closed-set attribution and authorship verification.

2 Related Work

Previous work in authorship attribution focuses mainly on stylometric features that capture aspects of personal writing style (Gamon, 2004; Luyckx and Daelemans, 2008; Escalante et al., 2011; Schwartz et al., 2013; Tschuggnall and Specht, 2014; Sidorov et al., 2014). In addition, beyond the use of typical classification algorithms, several attribution models that are specifically designed for authorship attribution tasks have been proposed (Koppel et al., 2011; Seroussi et al., 2014; Qian et al., 2014). Basic approaches and models are reviewed by Juola (2008) and Stamatatos (2009). In addition, recent studies in authorship verification are surveyed in (Stamatatos et al., 2014; Stamatatos et al., 2015).

An early cross-topic study in authorship attribution using a very small corpus (3 authors and 3 topics) showed that the identification of authors of email messages is not affected too much when the training and test messages are on different topics (de Vel et al., 2001). Based on another small corpus (2 authors and 3 topics) Madigan, *et al.* (2005) demonstrated that POS features are more effective than word unigrams in cross-topic conditions. The *unmasking* method for author verification of long documents based on very frequent word frequencies was successfully tested in cross-topic conditions (Koppel et al., 2007) but Kestemont, *et al.* (2012) found that its reliability was significantly lower in cross-genre conditions. Function words have been found to be effective when topics of the test corpus are excluded from the training corpus (Baayen et al., 2002; Goldstein-Stewart et al., 2009; Menon and Choi, 2011). However, Mikros and Argiri (2007) demonstrated that function word features actually correlate with topic. Other types of features found effective in cross-topic and cross-genre authorship attribution are punctuation mark frequencies (Baayen et al., 2002), LIWC features (Goldstein-Stewart et al., 2009), and character n -grams (Stamatatos, 2013). To enhance the performance of attribution models based on character n -gram features, Sapkota et al. (2015) define several n -gram categories and then they combine n -grams that correspond to word affixes and punctuation marks. Combining several topics in the training set seems also to enhance the ability to identify the authors of texts on another topic (Sapkota et al., 2014). More recently, Sapkota et al. (2016) proposed a domain adaptation model based on structural correspondence learning and punctuation-based character n -grams as pivot features.

Text distortion has successfully been used to enhance thematic text clustering by masking the occurrences of frequent words while maintaining the textual structure (Granados et al., 2011; Granados et al., 2012). That way, the clustering model was no longer confused by non relevant information hidden in the produced distorted text (Granados et al., 2014). An important conclusion drawn by these studies was that, in cases the textual structure was not maintained, the performance of clustering decreased despite the fact that the same thematic information was available.

3 Text Distortion Views

In this paper we propose a method to transform texts by applying a text distortion process before extracting the stylometric features. The main idea is to provide a new version of texts that is more topic-neutral in comparison to the original texts while maintaining most of the information related with the personal style of the author. Our method is inspired by the text distortion approach introduced by Granados, *et al.* (2011; 2012) but it significantly differs from that since it is more suitable for authorship attribution rather than thematic clustering. In more detail, given the k most frequent words of the language W_k , the proposed method transforms the input $Text$ as described in Algorithm 1.

Algorithm 1 DV-MA

Input: $Text, W_k$

Output: $Text$

- 1: Tokenize $Text$
 - 2: **for** each token t in $Text$ **do**
 - 3: **if** lowercase(t) $\notin W_k$ **then**
 - 4: replace each digit in t with #
 - 5: replace each letter in t with *
 - 6: **end if**
 - 7: **end for**
-

We call this method *Distorted View with Multiple Asterisks* (DV-MA). Alternatively, a single symbol can be used to replace sequences of digits/letters meaning that the token length information is lost. This version of the proposed method, called *Distorted View with Single Asterisks* (DV-SA) is illustrated in Algorithm 2.

Algorithm 2 DV-SA

Input: $Text, W_k$

Output: $Text$

- 1: Tokenize $Text$
 - 2: **for** each token t in $Text$ **do**
 - 3: **if** lowercase(t) $\notin W_k$ **then**
 - 4: replace any sequence of digits in t with a single #
 - 5: replace any sequence of letters in t with a single *
 - 6: **end if**
 - 7: **end for**
-

Note that DV-MA does not affect the length of input text while DV-SA reduces text-length. The

proposed text transformation is demonstrated in the example of Table 1 where an input text is transformed according to either DV-MA or DV-SA algorithms. In each example, W_k consists of the k most frequent words of the BNC corpus¹. As can be seen, each value of k provides a distorted view on the text where the textual structure is maintained but some, mainly thematically related, information is masked. In the extreme case where $k=0$ all words are replaced by asterisks and the only information left concerns word-length, punctuation marks and numbers usage. When $k=100$, function words remain visible and it is possible to extract patterns of their usage. Note that capitalization of letters remain unaffected. When k increases to include thousands of frequent words of BNC more topic-related information is visible. In general, the lower the k , the more thematic information is masked. By appropriately tuning parameter k , it is possible to decide how much thematic information is going to be compressed.

The text distortion method described in Granados, *et al.* (2011; 2012) has also been applied to the input text of Table 1 for $k=1,000$. In comparison to that method, the proposed approach is different in the following points:

- We replace the occurrences of the least frequent words rather than the most frequent words since it is well known that function word usage provide important stylometric information.
- Punctuation marks and other symbols are maintained since they are important style markers.
- Capitalization of original text is maintained.
- We treat numbers in a special way in order to keep them in the resulting text but in a more neutral way that reflects the stylistic choices of authors. For example, note that in each example of Table 1 both \$15,000 and \$17,000 are transformed to the same pattern. Thus, the proposed methods are able to capture the format used by the author and discard the non-relevant information about the exact numbers. In the case of DV-SA, any similar number (e.g., \$1,000, \$100,000) would have exactly the same transformation.

¹<https://www.kilgarriff.co.uk/bnc-readme.html>

- DV-SA: the input texts are distorted using the Algorithm 2.

5 Closed-set Attribution

5.1 Corpora

First, we examine the case where a given closed-set of candidate authors is given. Multiple corpora are nowadays available for this task. We selected to use the following two corpora:

1. CCAT-10: this is a subset of the *Reuters Corpus v.1* comprising 10 authors and 100 texts per author belonging to the CCAT thematic category (corporate and industrial news). This corpus has been used in several previous studies (Plakias and Stamatatos, 2008; Escalante et al., 2011; Sapkota et al., 2015). Separate training and test corpora of equal size are provided.
2. Guardian: this is a corpus of articles from *The Guardian* UK newspaper. It includes opinion articles by 13 authors on 4 thematic areas (politics, society, UK, and world) as well as book reviews by the same authors. It has been used in previous work that focused on authorship attribution in cross-topic and cross-genre conditions (Stamatatos, 2013; Sapkota et al., 2014). Following the practice of previous studies, we use at most 10 texts per author in each category of this corpus.

It is important to highlight the main difference among the above corpora. CCAT-10 authors tend to write newswire stories on specific subjects and this is consistent in both training and test corpora. On the other hand, in the Guardian corpus the texts by one author cover several thematic areas and two genres (opinion articles and book reviews). Therefore, it is expected that an authorship attribution method that is not robust to topic shifts will be less effective in the Guardian corpus. In CCAT-10, it is the combination of personal style and preferred thematic nuances that define each class (author).

To make this difference among the above corpora more clear, Table 2 shows the top fifteen words of each corpus with respect to their χ^2 value and a total frequency of at least five. As expected, most of these words correspond to named-entities and other topic-related information. For each word, the total term frequency tf , document frequency df (number of different documents where

CCAT-10				Guardian			
Word	tf	df	af	Word	tf	df	af
Prague	133	74	1	dam	14	3	1
crowns	168	43	1	technologies	6	2	1
ING	41	34	2	Congo	12	2	1
PX50	39	29	1	DRC	17	2	1
Wood	27	27	1	Rwandan	25	2	1
Patria	27	24	1	speakers	7	3	2
fixing	37	27	1	theft	8	3	2
Futures	23	21	1	columnist	6	2	2
Barings	58	22	2	enriched	6	2	2
pence	70	20	1	whatsoever	6	2	2
Banka	52	18	1	combatants	7	2	2
Petr	17	17	1	Gadafy	9	2	2
Czechs	49	17	1	wellbeing	9	2	2
Grenfell	48	17	1	Libya	21	2	2
derivatives	31	16	1	allusions	6	4	1

Table 2: Top fifteen words with respect to χ^2 in CCAT-10 and Guardian corpora together with occurrence statistics.

the word appears), and author frequency af (number of different authors in whose documents the word appears) are also provided. As can be seen, in CCAT-10 there are multiple words that are author-specific and tend to appear in multiple documents by that author (both tf and df are high). Thus, these words are useful indicators of authorship for that specific corpus. On the other hand, in the Guardian corpus, it is not easy to find words that appear in multiple documents of the same author and do not appear in documents by other authors (when af is low, df is also very low).

5.2 Attribution Model

From each text of the corpus (either in its original form or in the proposed distorted view) the stylometric features are extracted (either character n -grams or token n -grams) and then a SVM classifier with a linear kernel is built. Such a simple attribution model has been extensively used in previous work and proved to be an effective approach to closed-set authorship attribution (Plakias and Stamatatos, 2008; Stamatatos, 2013; Sapkota et al., 2014; Sapkota et al., 2015).

The BNC corpus is used to estimate the most frequent words of the English language. For each model, the appropriate parameter settings for n , f_t , and k are estimated based on grid search as described in Section 4.

5.3 Results

First, we apply the examined models to the CCAT-10 corpus. Based on 10-fold cross-validation on the training corpus we estimate the best parame-

		Acc.	n	f_t	k	N
Character n -grams	Baseline	80.6	6	15		18,859
	DV-MA	78.2	7	35	2,000	5,426
	DV-SA	77.4	7	35	4,000	5,708
Token n -grams	Baseline	80.0	1	20		1,805
	DV-MA	79.2	2	5	4,000	10,199
	DV-SA	79.4	2	10	4,000	4,023

Table 3: Accuracy results of closed-set attribution on the CCAT-10 corpus. For each model, parameter settings (n , f_t , k) and number of features (N) are also shown.

ter settings for each model. Then, the best models are applied to the test corpus. Table 3 presents the results of this experiment. As can be seen, the baseline models are the most effective ones using both character and token n -gram features. However, only the DV-SA model using character n -grams is statistically significantly worse according to a McNemar’s test with continuity correction ($p < 0.05$) (Dietterich, 1998). For character n -gram features, both the baseline models and the proposed models are based on long n -grams ($n=6$ or 7), longer than usually examined in authorship attribution studies. This reflects the topic-specificity of classes in that corpus since longer character n -grams are better able to capture thematic information. Moreover, the proposed distortion-based models were based on high values of k confirming that thematic information is important in that corpus. As concerns the token n -gram features, the baseline model was based on unigrams while bigrams were selected for the proposed methods.

Next, we applied the examined models on the opinion articles of the cross-topic Guardian corpus as follows: one thematic category was used as training corpus, another was used as validation corpus (to estimate the best parameter settings) and the remaining two categories were used as test corpus. Since there are four thematic categories in total, all 12 combinations were examined and the results are shown in Table 4. Here the results favour the proposed methods. Note that the average value of k is low indicating that most thematic information is masked. For character n -gram features, in almost all cases the distorted view models (both DV-MA and DV-SA) outperform the baseline. In many cases the difference with respect to the baseline is very high, especially for character n -gram models. According to a McNemar’s test with continuity correction ($p < 0.05$) on the overall performances, DV-MA based on character n -

grams is significantly better than all other models except the corresponding DV-SA model. The average k value of this model is very low (150) meaning that essentially only function words remain unmasked. It should be mentioned that the baseline character n -gram models in most cases are more effective than baseline token n -gram models. However, in average, they are worse than token n -grams due to their poor performance when the Society texts are used for training. This thematic category contains the least number of texts (Stamatatos, 2013). All examined models are based on significantly reduced feature sets in comparison to the CCAT-10 corpus indicating that in cross-topic conditions the least frequent features are not so useful.

In another experiment using the Guardian corpus, a cross-genre scenario was followed where the training and evaluation corpora come from different genres. In more detail, the book reviews category of that corpus was used as training corpus, one thematic category of opinion articles was used as validation corpus (to estimate the best parameter settings) and the remaining three thematic categories of opinion articles were used as test corpus. Again, all 4 combinations were examined (each time using a different validation corpus). Note that since the training and validation corpus belong to different genres we expect the attribution models to capture the cross-genre variation. What makes this experiment challenging is again the cross-topic variation since validation and test corpora do not share thematic properties. Table 5 presents the results for all tested models. Again, the proposed distortion-based models perform much better in comparison to the baseline models. In terms of overall performance, a McNemar’s test shows that DV-MA using character n -grams is significantly better ($p < 0.05$) than the rest of the models. Note also that the feature set size for most models in this experiment is further reduced in comparison to the previous experiment.

Tables 4 and 5 also show the average values of best parameter settings for the experiments related with the Guardian corpus. In comparison to the CCAT-10 corpus (Table 3), we see that shorter character n -grams are used for the Guardian corpus while parameter k is much smaller. All these reflect the cross-topic nature of this corpus. Note that the proposed method is able to take advantage of this fact by masking topic-specific information.

			Character n -grams			Token n -grams		
Train	Valid.	Test	Baseline	DV-MA	DV-SA	Baseline	DV-MA	DV-SA
P	S	U&W	83.1	86.0	87.4	78.3	77.8	79.2
P	U	S&W	84.4	89.9	89.4	73.7	82.7	81.0
P	W	S&U	88.8	88.8	85.5	83.6	85.5	84.9
S	P	U&W	34.0	44.4	46.4	48.5	48.5	48.5
S	U	P&W	32.1	47.7	47.2	49.1	45.8	49.1
S	W	P&U	35.4	48.8	49.0	50.5	53.6	51.6
U	P	S&W	69.8	80.7	69.3	68.7	72.1	65.4
U	S	P&W	71.6	69.1	72.5	67.7	67.2	65.1
U	W	P&S	76.4	82.2	79.3	75.9	70.7	75.9
W	P	S&U	71.7	84.9	82.9	71.1	80.9	78.9
W	S	P&U	70.8	87.8	88.6	68.3	77.2	79.2
W	U	P&S	76.4	91.0	90.8	73.6	85.1	82.8
Average Accuracy			66.2	75.1	74.0	67.4	70.6	70.1
n			3.8	4.1	4.1	1.2	1.1	1.4
f_t			33.3	27.5	30.4	32.9	30.4	30.4
k				541.7	283.3		425	425
N			3,665.5	1,649.8	1,722.8	528.7	382.6	403.7

Table 4: Accuracy results of closed-set attribution on the Guardian corpus in cross-topic conditions. P, S, U, and W correspond to Politics, Society, UK, and World thematic categories. In each row, best performance is in boldface. Average parameter settings and average number of features (N) are also given.

			Character n -grams			Token n -grams		
Train	Valid.	Test	Baseline	DV-MA	DV-SA	Baseline	DV-MA	DV-SA
B	P	S&U&W	38.3	60.1	58.1	43.9	49.8	51.4
B	S	P&U&W	41.0	57.7	52.0	47.7	49.7	50.0
B	U	P&S&W	39.3	57.4	49.6	40.1	48.5	51.8
B	W	P&S&U	40.5	59.1	59.5	48.0	50.8	52.4
Average Accuracy			39.4	58.9	55.8	44.0	49.7	51.9
n			4	3.8	4	1.8	1.5	2
f_t			45	41.3	35	32.5	35	31.3
k				150	800		1,650	775
N			1,563.8	694	1,082.8	184.3	451.3	330.3

Table 5: Accuracy results of closed-set attribution on the Guardian corpus in cross-genre conditions. B corresponds to book reviews while P, S, U, and W correspond to Politics, Society, UK, and World thematic categories of opinion articles. In each row, best performance is in boldface. Average parameter settings and average number of features (N) are also given.

5.4 Effect of Parameter k

So far, the parameter settings of the proposed models, as well as the baseline methods, were obtained using a validation corpus. To study the effect of the newly introduced parameter k , we performed an additional experiment this time using character n -gram features with fixed $n = 4$ and $f_t = 5$ and varying k values ($k \in \{100, 200, \dots, 1000, 1500, \dots, 5000\}$). Similar, for baseline models we used the same fixed n and f_t values. Figure 1 shows the performance of DV-MA, DV-SA, and baseline models on the CCAT-10 and Guardian corpora. Note that the results for CCAT-10 are directly comparable to Table 3. On the other hand, for the Guardian corpus we present the average performance of all possible 12 combinations (using one thematic category as training

corpus and another thematic category as test corpus). This is not directly comparable to Table 4 (where the test corpus of each case consists of two thematic categories).

As can be clearly seen in Figure 1 the effect of parameter k in DV-MA and DV-SA models is crucial. In the case of CCAT-10, performance in general increases with k reflecting the topic-specific information per author in this corpus. Despite the fact that the baseline approach is better than the proposed models in this corpus, a carefully selected k value (around 3,500) makes DV-MA equally effective to the baseline. On the other hand, in the Guardian cross-topic corpus, the proposed DV-MA and DV-SA models are better than the baseline for all examined k values. The best performance is obtained for low k and the accu-

racy decreases as k increases. This clearly shows that the use of topic-specific information in this corpus negatively affects the effectiveness of authorship attribution models. It is also notable that, in both corpora, the differences between DV-MA and DV-SA are not significant. However, DV-MA is slightly better than DV-SA in most of the cases.

6 Authorship Verification

6.1 Corpora

Recently, the PAN evaluation campaigns focused on the authorship verification task and several benchmark corpora were developed for this task (Stamatatos et al., 2014; Stamatatos et al., 2015). Each PAN corpus consists of a number of verification problems and each problem includes a set of known documents by the same author and exactly one document of unknown authorship. The task is to decide whether the known and the unknown documents are by the same author. In this paper, we use the following corpora:

PAN-2014: It includes 6 corpora covering 4 languages and several genres: Dutch essays (PAN14-DE), Dutch reviews (PAN14-DR), English essays (PAN14-EE), English novels (PAN14-EN), Greek newspaper articles (PAN14-GR) and Spanish newspaper articles (PAN14-SP). Each corpus is divided into training and test sets. Within each verification problem all documents belong to the same genre and fall into the same thematic area. Details of these corpora are presented in (Stamatatos et al., 2014).

PAN-2015: In this collection, within each verification problem the documents can belong to different genres and thematic areas. It includes a cross-genre corpus in Dutch (PAN15-DU), cross-topic corpora in English (PAN15-EN) and Greek (PAN15-GR) and a mixed (partially cross-topic and cross-genre) corpus in Spanish (PAN15-SP). More details of these corpora are provided in (Stamatatos et al., 2015).

6.2 Verification model

In this study, we use the authorship verification approach proposed by Potha and Stamatatos (2014). In more detail, this is a *profile-based* approach meaning that first it concatenates all available known documents and then it extracts a single representation from the resulting document (Stamatatos, 2009). The top L_k n -grams of the known text are then compared to the top L_u n -grams of

the unknown text and if the similarity is above a predefined threshold the unknown text is attributed to the author of the known texts. Note that parameters L_k and L_u essentially replace f_t that was used in previous experiments. All necessary parameters for this method, including n and k for the proposed method are estimated using the training part of each PAN corpus. Moreover, the most frequent words for each language are extracted from the corresponding training corpus.

6.3 Results

Table 6 shows the performance of the authorship verification models on the 10 PAN benchmark corpora based on the area under the Receiver-Operating-Characteristic curve (AUC-ROC). This evaluation measure was also used in PAN evaluation campaigns and the presented results can be directly compared to the ones reported by PAN organizers (Stamatatos et al., 2014; Stamatatos et al., 2015). In average, the proposed distortion-based models surpass the performance of the baseline models with both character n -gram and token n -gram features. The baseline model is better only in the case of the most challenging cross-genre PAN15-DU corpus. However, its performance essentially resembles random guessing (0.5). DV-SA models seem more competitive than DV-MA in the author verification task.

Table 6 also shows the performance of DV-Opt that corresponds to the best model (either DV-MA or DV-SA using either character or token n -grams) that can be selected by optimizing the performance (AUC-ROC) on the training corpus, separately for each one of the 10 corpora. The practice of using different models and settings per verification corpus is common in previous work (Seidman, 2013; Jankowska et al., 2014). As can be seen in Table 6, DV-Opt is better than any other single model in average performance and a one-tailed t -test shows that it is significantly better (at the 5% level) than both baseline models and DV-MA using character n -grams. The performance of DV-Opt is directly comparable to the results of PAN participants reported in (Stamatatos et al., 2014; Stamatatos et al., 2015) since the best models are selected based on information obtained from the training corpus. The last column of Table 6 compares DV-Opt with the overall winners of PAN-2014 (Khonji and Iraqi, 2014) and PAN-2015 (Bagnall, 2015). DV-Opt achieves better results in comparison to PAN

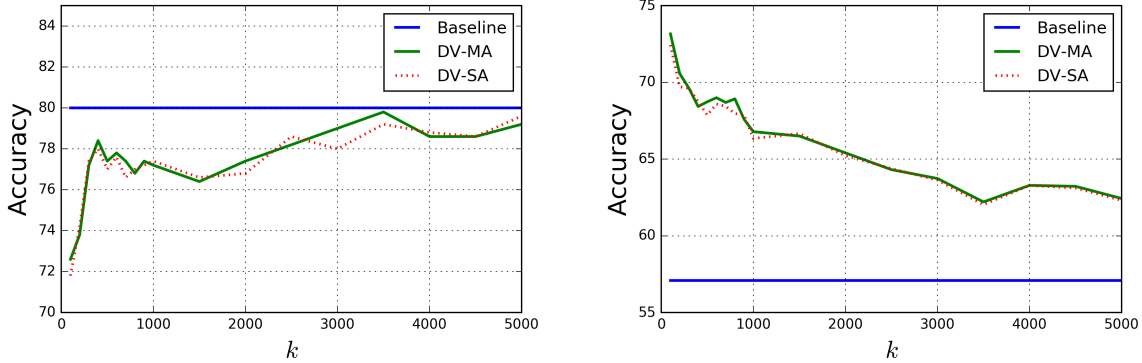


Figure 1: Performance of DV-MA, DV-SA, and baseline models based on character n -grams for fixed $n = 4$ and $f_t = 5$ and varying k values on CCAT-10 corpus (left) and the Guardian cross-topic corpus (right).

Corpus	Character n -grams			Token n -grams			DV-Opt	Diff. PAN Winner
	Baseline	DV-MA	DV-SA	Baseline	DV-MA	DV-SA		
PAN14-DE	0.975	0.961	0.979	0.948	0.919	0.900	0.961	+0.048
PAN14-DR	0.643	0.686	0.700	0.691	0.690	0.704	0.704	-0.032
PAN14-EE	0.528	0.591	0.582	0.626	0.690	0.606	0.690	+0.091
PAN14-EN	0.696	0.708	0.733	0.714	0.695	0.732	0.695	-0.055
PAN14-GR	0.625	0.783	0.779	0.794	0.838	0.853	0.783	-0.106
PAN14-SP	0.770	0.784	0.802	0.718	0.838	0.832	0.832	-0.066
PAN15-DU	0.519	0.470	0.509	0.247	0.433	0.505	0.509	-0.191
PAN15-EN	0.766	0.721	0.770	0.706	0.752	0.743	0.770	-0.041
PAN15-GR	0.710	0.720	0.706	0.672	0.674	0.646	0.720	-0.162
PAN15-SP	0.690	0.818	0.813	0.852	0.841	0.852	0.852	-0.034
Average	0.692	0.724	0.737	0.697	0.737	0.737	0.752	-0.055

Table 6: AUC-ROC scores of the examined authorship verification models. Last column shows the difference of DV-Opt with respect to the overall PAN-2014 and PAN-2015 winners.

winners in two corpora (PAN14-DE and PAN14-EE) while its performance is notably worse than PAN winners in PAN14-GR, PAN15-DU, and PAN15-GR. It should be underlined that the verification method used in this paper is an *intrinsic* approach while both PAN-2014 and PAN-2015 winners followed an *extrinsic* approach (where additional documents by other authors are considered in order to transform the verification problem to a binary classification task). Extrinsic models tend to perform better (Stamatatos et al., 2015).

7 Conclusions

In this paper, we presented techniques of text distortion that can significantly enhance the robustness of authorship attribution methods in challenging cases where the topic of documents by the same author varies. The proposed algorithms transform texts into a form where topic information is compressed while textual structure related to personal style is maintained. These algorithms are language-independent, do not require compli-

cated resources, and can easily be combined with existing authorship attribution methods. Experimental results demonstrated a considerable gain in effectiveness when using the proposed models under the realistic cross-topic conditions in both closed-set attribution and author verification tasks. On the other hand, when the corpora are too topic-specific where the texts by a given author are consistently on certain subjects different than the ones of the other candidate authors, the distortion methods seem not to be helpful. Parameter k can be carefully adjusted to reflect topic properties of a given corpus.

More experiments are needed in the case of cross-genre conditions to estimate if the proposed method is also able to compress genre information and at the same time maintain properties related to personal style of authors. It would also be interesting to examine whether the distorted views of texts can be useful to other style-based text categorization tasks, including author profiling and genre detection.

References

- A. Abbasi and Hsinchun Chen. 2005. Applying authorship analysis to extremist-group web forum messages. *Intelligent Systems, IEEE*, 20(5):67–75.
- Shlomo Argamon, Casey Whitelaw, Paul J. Chase, Sobhan Raj Hota, Navendu Garg, and Shlomo Levitan. 2007. Stylistic text classification using functional lexical features. *Journal of the American Society for Information Science and Technology*, 58(6):802–822.
- Harald Baayen, Hans van Halteren, Anneke Neijt, and Fiona Tweedie. 2002. An experiment in authorship attribution. In *6th JADT*, pages 29–37.
- Douglas Bagnall. 2015. Author Identification using multi-headed Recurrent Neural Networks. In L. Cappellato, N. Ferro, J. Gareth, and E. San Juan, editors, *Working Notes Papers of the CLEF 2015 Evaluation Labs*. CLEF and CEUR-WS.org.
- Malcolm Coulthard. 2013. On admissible linguistic evidence. *Journal of Law and Policy*, XXI(2):441–466.
- Olivier Y. de Vel, Alison Anderson, Malcolm Corney, and George M. Mohay. 2001. Mining email content for author identification forensics. *SIGMOD Record*, 30(4):55–64.
- Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.
- Hugo Jair Escalante, Tamar Solorio, and Manuel Montes-y Gomez. 2011. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 288–298, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Michael Gamon. 2004. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In *Proceedings of Coling 2004*, pages 611–617, Geneva, Switzerland, Aug 23–Aug 27. COLING.
- Jade Goldstein-Stewart, Ransom Winder, and Roberta Sabin. 2009. Person identification from text and speech genre samples. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 336–344, Athens, Greece, March. Association for Computational Linguistics.
- Ana Granados, Manuel Cebrián, David Camacho, and Francisco de Borja Rodríguez. 2011. Reducing the loss of information through annealing text distortion. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1090–1102.
- Ana Granados, David Camacho, and Francisco de Borja Rodríguez. 2012. Is the contextual information relevant in text clustering by compression? *Expert Systems with Applications*, 39(10):8537–8546.
- Ana Granados, Rafael Martínez, David Camacho, and Francisco de Borja Rodríguez. 2014. Improving NCD accuracy by combining document segmentation and document distortion. *Knowledge and Information Systems*, 41(1):223–245.
- Jack Grieve. 2007. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing*, 22(3):251–270.
- Steffen Hedegaard and Jakob Grue Simonsen. 2011. Lost in translation: Authorship attribution using frame semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 65–70, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Magdalena Jankowska, Evangelos Milios, and Vlado Keselj. 2014. Author verification using common n-gram profiles of text documents. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 387–397, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.
- Patrick Juola. 2008. Authorship Attribution. *Foundations and Trends in Information Retrieval*, 1:234–334.
- Patrick Juola. 2013. How a computer program helped reveal J. K. Rowling as author of A Cuckoo’s Calling. *Scientific American*.
- Mike Kestemont, Kim Luyckx, Walter Daelemans, and Thomas Crombez. 2012. Cross-genre authorship verification using unmasking. *English Studies*, 93(3):340–356.
- Mahmoud Khonji and Youssef Iraqi. 2014. A slightly-modified GI-based author-verifier with lots of features (ASGALF). In *CLEF 2014 Labs and Workshops, Notebook Papers*. CLEF and CEUR-WS.org.
- Moshe Koppel and Shachar Seidman. 2013. Automatically identifying pseudepigraphic texts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1449–1454, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the American Society for Information Science and Technology*, 65(1):178–187.
- Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8:1261–1276.

- Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94.
- Maarten Lambers and Cor J. Veenman. 2009. Forensic authorship attribution using compression distances to prototypes. In Zeno J.M.H. Geradts, Katrin Y. Franke, and Cor J. Veenman, editors, *Computational Forensics: Third International Workshop*, volume 5718 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg.
- Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 513–520, Manchester, UK, August. Coling 2008 Organizing Committee.
- David Madigan, Alexander Genkin, David D Lewis, Shlomo Argamon, Dmitriy Fradkin, and Li Ye. 2005. Author identification on the large scale. In *Proceedings of the Meeting of the Classification Society of North America*.
- Rohith Menon and Yejin Choi. 2011. Domain independent authorship attribution without domain adaptation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 309–315, Hissar, Bulgaria, September. RANLP 2011 Organising Committee.
- George K. Mikros and Eleni K. Argiri. 2007. Investigating topic influence in authorship attribution. In *Proceedings of the SIGIR 2007 International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, PAN*.
- Spyridon Plakias and Efstathios Stamatatos. 2008. Author identification using a tensor space representation. In *Proceedings of the 18th European Conference on Artificial Intelligence ECAI*, pages 833–834.
- Nektaria Potha and Efstathios Stamatatos. 2014. A profile-based method for authorship verification. In *Artificial Intelligence: Methods and Applications - Proceedings of the 8th Hellenic Conference on AI, SETN*, pages 313–326.
- Tieyun Qian, Bing Liu, Li Chen, and Zhiyong Peng. 2014. Tri-training for authorship attribution with limited training data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 345–351, Baltimore, Maryland, June. Association for Computational Linguistics.
- Upendra Sapkota, Thamar Solorio, Manuel Montes, Steven Bethard, and Paolo Rosso. 2014. Cross-topic authorship attribution: Will out-of-topic data help? In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1228–1237, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.
- Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, Denver, Colorado, May–June. Association for Computational Linguistics.
- Upendra Sapkota, Thamar Solorio, Manuel Montes, and Steven Bethard. 2016. Domain adaptation for authorship attribution: Improved structural correspondence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2226–2235, Berlin, Germany, August. Association for Computational Linguistics.
- Jacques Savoy. 2013. Authorship attribution based on a probabilistic topic model. *Information Processing and Management*, 49(1):341–354.
- Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Shachar Seidman. 2013. Authorship verification using the impostors method notebook for PAN at CLEF 2013. In *Working Notes for CLEF 2013 Conference*.
- Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship attribution with topic models. *Computational Linguistics*, 40(2):269–310.
- Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. 2014. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860.
- Efstathios Stamatatos, Nikos Fakotakis, and George Kokkinakis. 2000. Automatic text categorization in terms of genre and author. *Computational Linguistics*, 26(4):471–495.
- Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Benno Stein, Martin Potthast, Patrick Juola, Miguel A. Sánchez-Pérez, and Alberto Barrón-Cedeño. 2014. Overview of the author identification task at PAN 2014. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, pages 877–897.
- Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, and Benno Stein. 2015. Overview of the author identification task at PAN 2015. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015*.

- Efstathios Stamatatos. 2007. Author identification using imbalanced and limited training texts. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications, DEXA '07*, pages 237–241, Washington, DC, USA. IEEE Computer Society.
- Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60:538–556.
- Efstathios Stamatatos. 2013. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21:421–439.
- Justin Anthony Stover, Yaron Winter, Moshe Koppel, and Mike Kestemont. 2016. Computational authorship verification method attributes a new work to a major 2nd century african author. *Journal of the American Society for Information Science and Technology*, 67(1):239–242.
- Michael Tschuggnall and Günther Specht. 2014. Enhancing authorship attribution by utilizing syntax tree profiles. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 195–199, Gothenburg, Sweden, April. Association for Computational Linguistics.
- Hans van Halteren. 2004. Linguistic profiling for authorship recognition and verification. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 199–206, Barcelona, Spain, July.

Structured Learning for Temporal Relation Extraction from Clinical Records

Artuur Leeuwenberg and Marie-Francine Moens

Department of Computer Science

KU Leuven, Belgium

{tuur.leeuwenberg, sien.moens}@cs.kuleuven.be

Abstract

We propose a scalable structured learning model that jointly predicts temporal relations between events and temporal expressions (TLINKS), and the relation between these events and the document creation time (DCTR). We employ a structured perceptron, together with integer linear programming constraints for document-level inference during training and prediction to exploit relational properties of temporality, together with global learning of the relations at the document level. Moreover, this study gives insights in the results of integrating constraints for temporal relation extraction when using structured learning and prediction. Our best system outperforms the state-of-the-art on both the CONTAINS TLINK task, and the DCTR task.

1 Introduction

Temporal information is critical in many clinical areas (Combi and Shahar, 1997). A big part of this temporal information is captured in the free text of patient records. The current work aims to improve temporal information extraction from such clinical texts. Extraction of temporal information from clinical text records can be used to construct a time-line of the patient's condition (such as in Figure 1). The extracted time-line can help clinical researchers to better select and recruit patients with a certain history for clinical trials. Moreover, the time-line is crucial for making a good patient prognosis and clinical decision support (Onisko et al., 2015; Stacey and McGregor, 2007).

Temporal information extraction can be divided into three sub-problems: (1) the detection of events (E_e); (2) the detection of temporal expressions (E_t); and (3) the detection of temporal rela-

tions between them. In the clinical domain, events include medical procedures, treatments, or symptoms (e.g. *colonoscopy*, *smoking*, *CT-scan*). Temporal expressions include dates, days of the week, months, or relative expressions like *yesterday*, *last week*, or *post-operative*. In this work, we focus on the last sub-problem, extraction of temporal relations (assuming events and temporal expressions are given). As a small example of the task we aim to solve, given the following sentence:

In 1990 the patient was diagnosed and received surgery directly afterwards.

in which we assume that the events *diagnosed* and *adenocarcinoma*, and the temporal expression *1990* are given, we wish to extract the following relations:

- CONTAINS(1990, diagnosed)
- CONTAINS(1990, surgery)
- BEFORE(diagnosed, surgery)
- BEFORE(diagnosed, d)
- BEFORE(surgery, d)

where d stands for the document creation time.

Our work leads to the following contributions: First, we propose a scalable structured learning model that jointly predicts temporal relations between events and temporal expressions (TLINKS), and the relation between these events and the document creation time (DCTR). In contrast to existing approaches which detect relation instances separately, our approach employs a structured perceptron (Collins, 2002) for global learning with joint inference of the temporal relations on a document level. Second, we ensure scalability through using integer linear programming (ILP)

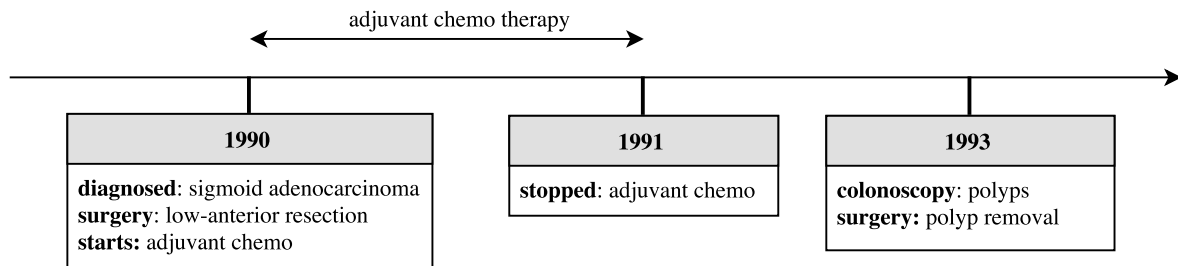


Figure 1: Fragment of a (partial) patient time-line.

constraints with fast solvers, loss augmented sub-sampling, and good initialization. Third, this study leads to valuable insights on when and how to make inferences over the found candidate relations both during training and prediction and gives an in-depth assessment of the use of additional constraints and global features during inference. Finally, our best system outperforms the state-of-the-art of both the CONTAINS TLINK task, and the DCTR task.

2 Related Work

There have been two shared tasks on the topic of temporal relation extraction in the clinical domain: the I2B2 Temporal Challenge (Sun et al., 2013), and more recently the Clinical TempEval Shared Task with two iterations, one in 2015 and one in 2016 (Bethard et al., 2014; Bethard et al., 2015; Bethard et al., 2016). In the I2B2 Temporal Challenge eight types of relations were initially annotated. However, due to low inter-annotator agreement these were merged to three types of temporal relations, OVERLAP, BEFORE, and AFTER. Good annotation of temporal relations is difficult, as annotators frequently miss relation mentions. In the Clinical TempEval Shared tasks the THYME corpus is used (Styler IV et al., 2014), with a different annotation scheme that aims at annotating those relations that are most informative w.r.t. the time-line, and gives less priority to relations that can be inferred from the others. This results in two categories of temporal relations: The relation between each event and the document creation time (DCTR), dividing all events in four temporal buckets (BEFORE, BEFORE/OVERLAP, OVERLAP, AFTER). These buckets are called narrative containers (Pustejovsky and Stubbs, 2011). And second, relations between temporal entities that both occur in the text (TLINKS). TLINKS may occur between events ($E_e \times E_e$), and between events

and temporal expressions ($E_e \times E_t$ and $E_t \times E_e$). The TLINK types (and their relative frequency in the THYME corpus) are CONTAINS (64,42%), OVERLAP (15,19%), BEFORE (12,65%), BEGINS-ON (6.15%), and ENDS-ON (1.59%). The relations AFTER, and DURING are expressed in terms of their inverse, BEFORE, and CONTAINS respectively. In our experiments, we use the THYME corpus for its relatively high inter-annotator agreement (particularly for CONTAINS).

To our knowledge, in all submissions (4 in 2015, and 10 in 2016) of Clinical TempEval the task is approached as a classical entity-relation extraction problem, and the predictions for both categories of relations are made independently from each other, or in a one way dependency, where the containment classifier uses information about the predicted document-time relation. Narrative containment, temporal order, and document-time relation have very strong dependencies. Not modeling these may result in inconsistent output labels, that do not result in a consistent time-line.

An example of inconsistent labeling is given in Figure 2. The example is inconsistent when assigning the AFTER label for the relation between *lesion* and the document-time. It is inconsistent because we can also infer that *lesion* occurs BEFORE the document-time, as the *colonoscopy* event occurs before the document-time, and the *lesion* is contained by the *colonoscopy*.

Temporal inference, in particular *temporal closure*, is frequently used to expand the training data (Mani et al., 2006; Chambers and Jurafsky, 2008; Lee et al., 2016; Lin et al., 2016b), most of the times resulting in an increase in performance, and is also taken into account when evaluating the predicted labels (Bethard et al., 2014; UzZaman and Allen, 2011). Only very limited research regards the modeling of temporal dependencies into the machine learning model. (Chambers and Juraf-

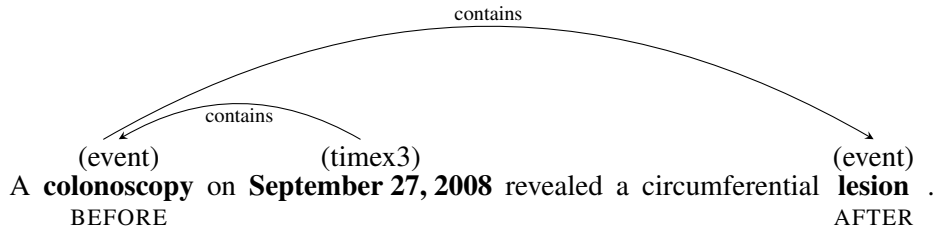


Figure 2: Example of inconsistent output labeling. Containment is indicated by directed edges, and the relation to the document-time by small caps below the events.

sky, 2008) and (Do et al., 2012) modeled label dependencies when predicting TimeBank TLINKS (Pustejovsky et al., 2003). They trained local classifiers and used a set of global temporal label constraints. Integer linear programming was employed to maximize the score from the local classifiers, while satisfying the global label constraints at prediction time. For both, this gave a significant increase in performance, and resulted in consistent output labels.

(Yoshikawa et al., 2009) modeled the label dependencies between TLINKS and DCTR with Markov Logic Networks (MLN), allowing for soft label constraints during training and prediction. However, MLN can sometimes be sub-optimal for text mining tasks w.r.t. time efficiency (Mojica and Ng, 2016). Quite recently, for a similar problem, spatial relation extraction, (Kordjamshidi et al., 2015) used an efficient combination of a structured perceptron or structured support vector machine with integer linear programming. In their experiments, they compare a local learning model (LO), a local learning model with global inference at prediction time (L+I), and a structured learning model with and without inference during training (IBT+I, and IBT-I respectively). In their experiments L+I gave better results than LO, but a more significant improvement was made when using structured learning in contrast to local learning.

In this work, we aim to jointly predict TLINKS and DCTR in a structured learning model with inference during training and prediction, to assess inference with temporal constraints of (Chambers and Jurafsky, 2008; Do et al., 2012) for the THYME relations, and to experiment with both local, and document-level inference for temporal information extraction in the clinical domain.

3 The Model

For jointly learning both tasks on a document level, we employ a structured perceptron learning paradigm (Collins, 2002). The structured perceptron model uses a joint feature function $\Phi(X, Y)$ to represent a full input document X with a label assignment Y . During training the model learns a weight vector λ to score how good the label assignment is. Predicting label assignment Y for a document X corresponds to finding the Y with the maximal score. In the following sub-sections we define the joint feature function Φ , describe the prediction procedure of the model, and provide how we train the model (i.e. learn a good λ).

3.1 Joint Features

To compose the joint feature function, we first define two local feature functions: $\phi_{tl} : (x, y) \rightarrow \mathbb{R}^p$ assigns features for the local classifications regarding TLINKS (with possible labels $L_{tl} = \{\text{CONTAINS, BEFORE, OVERLAP, BEGINS-ON, ENDS-ON, NO_LABEL}\}$), and a second local feature function $\phi_{dr} : (x, y) \rightarrow \mathbb{R}^q$, for local features regarding document-time relation classification (with labels $L_{dr} = \{\text{BEFORE, BEFORE_OVERLAP, OVERLAP, AFTER}\}$). The features used by these local feature functions are given in Table 1.

From these, we define a joint feature function $\Phi_{joint} : (X, Y) \rightarrow \mathbb{R}^{p+q}$, that concatenates (\oplus) the summed local feature vectors, creating the feature vector for the global prediction task (predicting all labels in the document for both sub-tasks at once). Φ_{joint} is defined in Equation 1, where $C_{tl}(X)$ and $C_{dr}(X)$ are candidate generation functions for the TLINK sub-task, and DCTR sub-task respectively (further explained in Section 3.2).

Features	ϕ_{dr}	ϕ_{tl}
String features for tokens and POS of each entity	✓	✓
String features for tokens and POS in a window of size $\{3, 5\}$, left and right of each entity	✓	✓
Boolean features for entity attributes (event polarity, event modality, event degree, and type)	✓	✓
String feature for the token and POS of the closest verb	✓	
String feature for the token and POS of the closest left and right entity	✓	
String features for the token $\{1, 2, 3\}$ -grams and POS $\{1, 2, 3\}$ -grams in-between the two entities		✓
Dependency path between entities (consisting of POS and edge labels)		✓
Boolean feature on if the first argument occurs before the second (w.r.t. word order)		✓

Table 1: Features of the local feature functions of each sub-task, ϕ_{tl} for TLINKS, and ϕ_{dr} for DCTR.

$$\Phi_{joint}(X, Y) = \sum_{x \in C_{dr}(X)} \sum_{l \in L_{dr}} \phi_{dr}(x, l) \oplus \sum_{x \in C_{tl}(X)} \sum_{l \in L_{tl}} \phi_{tl}(x, l) \quad (1)$$

3.2 Local Candidate Generation

For each document X , we create local candidates for both sub-tasks. In this work, we assume that event (E_e) and temporal expression (E_t) annotations are provided in the input. The DCTR-candidates in document X are then given by $C_{dr}(X)$, which returns all events in the document, i.e. $E_e(X)$. $C_{tl}(X)$ returns all TLINK candidates, i.e. $E_e(X) \cup E_t(X) \times E_e(X)$. In our experiments we usually restrict the number of candidates generated by C_{tl} to gain training and prediction speed (without significant loss in performance). This is explained further in Section 4.3.

3.3 Global Features

We also experiment with a set of global features, by which we mean features that are expressed in terms of multiple local labels. The global features are specified in Table 2. Global features are defined by a feature function $\Phi_{global}(X, Y) \rightarrow \mathbb{R}^r$ and have their corresponding weights in weight vector λ . When using global features Φ_{global} is concatenated with the joint feature function Φ_{joint} to form the final feature function Φ , as shown in Equation 2.

$$\Phi(X, Y) = \Phi_{joint}(X, Y) \oplus \Phi_{global}(X, Y) \quad (2)$$

When not using global features, we use only the joint features, as shown in Equation 3.

$$\Phi(X, Y) = \Phi_{joint}(X, Y) \quad (3)$$

Feature	Description
Φ_{sdr}	Bigram and trigram counts of subsequent DCTR-labels in the document
Φ_{drtl}	Counts of DCTR-label pairs of the entities of each TLINK

Table 2: Global (document-level) features.

3.4 Prediction

The model assigns a score to each input document X together with output labeling Y . The score for (X, Y) is defined as the dot product between the learned weight vector λ and the outcome of the joint feature function $\Phi(X, Y)$, as shown in Equation 4.

$$S(X, Y) = \lambda \Phi(X, Y) \quad (4)$$

The prediction problem for an input document X is finding the label assignment Y that maximizes the score S based on the weight vector λ , shown in Equation 5.

$$\hat{Y}_k = \arg \max_Y S(X, Y) \quad (5)$$

We use integer linear programming (ILP) to solve the prediction problem in Equation 5. Each possible local decision is modeled with a binary decision variable. For each local relation candidate input $x_{i,j}$ (for the relation between i and j) a binary decision variable $w_{i,j}^l$ is used for each potential label l that could be assigned to $x_{i,j}$, depending on the sub-task. The objective of the integer linear program, given in Equation 6, is to maximize the sum of the scores of local decisions. In all equations the constant d refers to the document-creation time. The objective is maximized under two sets of constraints, given in Equations 7 and 8, that express that each candidate is assigned ex-

actly one label, for each sub-task.

$$O = \arg \max_W \sum_{x_{i,d} \in C_{dr}(X)} \sum_{l \in L_{dr}} w_{i,d}^l \cdot S(x_{i,d}, y_{i,d}^l) + \sum_{x_{i,j} \in C_{tl}(X)} \sum_{l \in L_{tl}} w_{i,j}^l \cdot S(x_{i,j}, y_{i,j}^l) \quad (6)$$

$$\forall_i : \sum_{l \in L_{dr}} w_{i,d}^l = 1 \quad (7)$$

$$\forall_{i,j} : \sum_{l \in L_{tl}} w_{i,j}^l = 1 \quad (8)$$

For solving the integer linear program we use Gurobi (Gurobi Optimization, 2015).

3.4.1 Temporal Label Constraints

Because temporal relations are interdependent, we experimented with using additional constraints on the output labeling. The additional temporal constraints we experiment with are shown in Table 3. Constraints are expressed in terms of the binary decision variables used in the integer linear program.

In Table 3, constraints \mathcal{C}_{Ctrans} , and \mathcal{C}_{Btrans} model transitivity of CONTAINS, and BEFORE respectively. Constraints \mathcal{C}_{CBB} , and \mathcal{C}_{CAA} model the consistency between TLINK relation CONTAINS and DCTR relations BEFORE, and AFTER respectively (resolving the inconsistent example of \mathcal{C}_{CBB} in section 1, and Figure 2). Similarly, \mathcal{C}_{BBB} , and \mathcal{C}_{BAA} model the consistency between TLINK relation BEFORE and DCTR relations BEFORE, and AFTER.

Constraints can be applied during training and prediction, as Equation 5 is to be solved for both. If not mentioned otherwise, we use constraints both during training and prediction.

3.5 Training

The training procedure for the averaged structured perceptron is given by Algorithm 1, for I iterations, on a set of training documents T . Notice that the prediction problem is also present during training, in line 6 of the algorithm. Weight vector λ is usually initialized with ones, and updated when the predicted label assignment \hat{Y}_k for input document X_k is not completely correct. The structured perceptron training may suffer from overfitting. Averaging the weights over the training examples of each iteration is a commonly used way to counteract this handicap (Collins, 2002; Freund

and Schapire, 1999). In Algorithm 1, c is used to count the number of training updates, and λ_a as a cache for averaging the weights. We also employ local loss-augmented negative sub-sampling, and local pre-learning to address class-imbalance and training time.

Algorithm 1 Averaged Structured Perceptron

Require: $\lambda, \lambda_a, c, I, T$
1: $c \leftarrow 0$
2: $\lambda \leftarrow \langle 1, \dots, 1 \rangle$
3: $\lambda_a \leftarrow \langle 1, \dots, 1 \rangle$
4: **for** i in I **do**
5: **for** k in T **do**
6: $\hat{Y}_k \leftarrow \arg \max_Y \lambda \Phi(X_k, Y)$
7: **if** $\hat{Y}_k \neq Y_k$ **then**
8: $\lambda \leftarrow \lambda + \Phi(X_k, Y_k) - \Phi(X_k, \hat{Y}_k)$
9: $\lambda_a \leftarrow \lambda_a + c \cdot \Phi(X_k, Y_k) - c \cdot \Phi(X_k, \hat{Y}_k)$
10: $c \leftarrow c + 1$
return $\lambda - \lambda_a / c$

3.5.1 Loss-augmented Negative Sub-sampling

For the TLINK sub-task, we have a very large negative class (NO_LABEL) and a relatively small positive class (the other TLINK labels) of training examples. To speed up training convergence and address class imbalance at the same time, we sub-sample negative examples during training. Within a document X , for each positive local training example $(x_{positive}, y_{positive})$ we take 10 random negative examples and add the negative example $(x_{negative}, y_{no_label})$ with the highest score for relation $y_{positive}$, i.e. $S(x_{negative}, y_{positive})$. This cutting plane optimization gives preference to negative training examples that are more likely to be classified wrongly, and thus can be learned from (in an online manner), and it provides only one negative training example for each positive training example, balancing the TLINK classes.

3.5.2 Local Initialization

To reduce training time, we don't initialize λ with ones, but we train a perceptron for both local sub-tasks, based on the same local features mentioned in Table 1, and use the trained weights to initialize λ for those features. A similar approach was used by (Weiss et al., 2015) for dependency parsing. Details on the training parameters of the perceptron are given in Section 4.3.

4 Experiments

We use our experiments to look at the effects of four modeling settings.

Abbrev.	Label Dependencies	Constraints
\mathcal{C}_{Ctrans}	$\text{CONTAINS}_{i,j} \wedge \text{CONTAINS}_{j,k} \rightarrow \text{CONTAINS}_{i,k}$	$\forall_{i,j,k} : w_{i,k}^{\text{contains}} - w_{i,j}^{\text{contains}} - w_{j,k}^{\text{contains}} \geq -1$
\mathcal{C}_{Btrans}	$\text{BEFORE}_{i,j} \wedge \text{BEFORE}_{j,k} \rightarrow \text{BEFORE}_{i,k}$	$\forall_{i,j,k} : w_{i,k}^{\text{before}} - w_{i,j}^{\text{before}} - w_{j,k}^{\text{before}} \geq -1$
\mathcal{C}_{CBB}	$\text{CONTAINS}_{i,j} \wedge \text{BEFORE}_{i,d} \rightarrow \text{BEFORE}_{j,d}$	$\forall_{i,j} : w_{j,d}^{\text{before}} - w_{i,j}^{\text{contains}} - w_{i,d}^{\text{before}} \geq -1$
\mathcal{C}_{CAA}	$\text{CONTAINS}_{i,j} \wedge \text{AFTER}_{i,d} \rightarrow \text{AFTER}_{j,d}$	$\forall_{i,j} : w_{j,d}^{\text{after}} - w_{i,j}^{\text{contains}} - w_{i,d}^{\text{after}} \geq -1$
\mathcal{C}_{BBB}	$\text{BEFORE}_{i,j} \wedge \text{BEFORE}_{j,d} \rightarrow \text{BEFORE}_{i,d}$	$\forall_{i,j} : w_{i,d}^{\text{before}} - w_{i,j}^{\text{before}} - w_{j,d}^{\text{before}} \geq -1$
\mathcal{C}_{BAA}	$\text{BEFORE}_{i,j} \wedge \text{AFTER}_{i,d} \rightarrow \text{AFTER}_{j,d}$	$\forall_{i,j} : w_{j,d}^{\text{after}} - w_{i,j}^{\text{before}} - w_{i,d}^{\text{after}} \geq -1$

Table 3: Temporal label dependencies expressed as integer linear programming constraints. The variables i, j and k range over the corresponding TLINK arguments, and constant d refers to the document-creation-time. $\text{CONTAINS}_{i,j}$ indicates that entity i contains entity j .

1. Document-level learning in contrast to pairwise entity-relation learning.
2. Joint learning of DCTR and TLINKS.
3. Integrating temporal label constraints.
4. Using global structured features.

We will discuss our results in Section 4.4. But first, we describe how we evaluate our system, and provide information on our baselines, and the pre-processing and hyper-parameter settings used in the experiments.

4.1 Evaluation

We evaluate our method on the clinical notes test set of the THYME corpus (Styler IV et al., 2014), also used in the Clinical TempEval 2016 Shared Task (Bethard et al., 2016). Some statistics about the dataset can be found in Table 4. F-measure is used as evaluation metric. For this we use the evaluation script from the Clinical TempEval 2016 Shared Task. TLINKS are evaluated under the temporal closure (Uzzaman and Allen, 2011).

Section	Documents	TLINKS	EVENTS
Train	440	17.109	38.872
Test	151	8.903	18.989

Table 4: Dataset statistics for the THYME sections we used in our experiments.

4.2 Baselines

Our first baseline is a perceptron algorithm, trained for each local task using the same local features as used to compose the joint feature function Φ_{joint} of our structured perceptron. We have

two competitive state-of-the-art baselines, one for the DCTR sub-task, and one for the TLINK sub-task. The first baseline is the best performing system of the Clinical TempEval 2016 on the DCTR task (Khalifa et al., 2016). They experiment with a feature rich SVM and a sequential conditional random field (CRF) for the prediction of DCTR and report the – to our knowledge – highest performance on the DCTR task. The competitive TLINK baseline is the latest version of the cTAKES Temporal system (Lin et al., 2016b; Lin et al., 2016a). They employ two SVMs to predict TLINKS, one for TLINKS between events, and one for TLINKS between events and temporal expressions and recently improved their system by generating extra training data using extracted UMLS concepts. They report the – to our knowledge – highest performance on CONTAINS TLINKS in the THYME corpus.

4.3 Hyper-parameters and Preprocessing

In all experiments, we preprocess the text by using a very simple tokenization procedure considering punctuation¹ or newline tokens as individual tokens, and splitting on spaces. For our part-of-speech (POS) features, and dependency parse path features, we rely on the cTAKES POS tagger and cTAKES dependency parser respectively (Savova et al., 2010). After POS tagging and parsing we lowercase the tokens. As mentioned in Section 3.2, we restrict our TLINK candidate generation in two ways. First, both entities should occur in a token window of 30, selected from $\{20, 25, 30, 35, 40\}$ based on development set performance. And second, both entities should occur in the same paragraph (paragraphs are separated

¹,./\ "' =+ - ; : () ! ? < > % & \$ * [] { }

by two consecutive newlines). Our motivation for not using sentence based candidate generation is that the clinical records contain many ungrammatical phrases, bullet point enumerations, and tables that may result in missing cross-sentence relation instances (Leeuwenberg and Moens, 2016). In all experiments, we train the normal perceptron for 8 iterations, and the structured perceptron for 32 iterations, both selected from $\{1, 2, 4, 8, 16, 32, 64\}$ based on best performance on the development set. The baseline perceptron is also used for the initialization of the structured perceptron. Moreover, we apply the transitive closure of CONTAINS, and BEFORE on the training data.

4.4 Results

Our experimental results on the THYME test set are reported in Table 5. In the table, the abbreviation SP refers to the structured perceptron model described in Section 3 but without temporal label constraints or global features, i.e. the joint document-level unconstrained structured perceptron, using local initialization, and loss-augmented negative sub-sampling. We compare this model with a number of modified versions to explore the effect of the modifications.

4.4.1 Document-Level Learning

When we compare the local perceptron baseline with any of the document-level models (any SP variation), we can clearly see that learning the relations at a document-level improves our model significantly² ($P < 0.0001$ for both DCTR and TLINKS). Furthermore, when comparing loss-augmented sub-sampling (SP) with random sub-sampling of negative TLINKS (SP_{random sub-sampling}) it can be seen that a good selection of negative training instances is very important for learning a good model (again $P < 0.0001$), and resulted in our model to improve the state-of-the-art by 1.4 on the CONTAINS TLINK task³.

4.4.2 Jointly Learning DCTR and TLINKS

When comparing the disjoint model (SP_{disjoint}) with our joint model (SP) it can be noticed that joint prediction gives only a very small improvement ($P = 0.0768$ for TLINKS, and $P = 0.0451$ for

²Significance is based on a document-level paired t-test.

³Only CONTAINS is generally reported for the THYME corpus, as the other TLINKS are less frequent, and the inter-annotator agreement for them is very low. We included them just for completeness in our experiments.

DCTR). However, joint learning on a document level provides the flexibility to formulate constraints connecting the labels of both tasks, such as the last four constraints in Table 3, resulting in a more consistent labeling over both tasks. Similarly, in the joint learning setting, we can define global features that connect both tasks (like Φ_{drtl}).

4.4.3 Integrating Temporal Constraints

We experimented with integrating label constraints in two ways (1) both during training and prediction (SP^{cc} + \mathcal{C}_*), or (2) only during prediction (SP^{uc} + \mathcal{C}_*). In general it can be noticed that in our experiments using the temporal label constraints from Table 3 slightly increases DCTR performance, but slightly decreases TLINK performance. A reason for this can be that the model generally gives better predictions for DCTR, that might result in providing a better alternative to a constraint violating solution. A difference in consistency of the annotation between both tasks could also be a reason. Furthermore, we can see that integrating the constraints both during training and prediction gives slightly lower performance compared only integrating them during prediction.

4.4.4 Using Global Structured Features

We have two types of features, Φ_{sdr} , which is only based on DCTR labels, and Φ_{drtl} , which is based on a combination of DCTR and TLINK labels. When we add Φ_{sdr} to our model, the overall F-measure on the DCTR task improves with 1.3 points ($P < 0.0001$), improving the state-of-the-art by 0.3 points. A reason for this can be the sequential dependency of DCTR labels, also exploited by (Khalifa et al., 2016), using the sequential CRF. The second global feature, Φ_{drtl} , in fact models the same type of dependencies as the last four constraints in Table 3, relating the TLINK relations with the DCTR labels of each TLINK argument, however as a soft dependency and not as a hard constraint. In our experiments, this feature did not improve either of the two sub-tasks. It appears that training with cross-task constraints, or global cross-task features is not trivial, and further research is needed on how to exploit these cross-task dependencies also during training. We assume that the lower-than-expected scores when modeling cross-task dependencies may be related to sub-sampling the negative TLINK training instances.

System	$F_{\text{BEFORE}}^{\text{DCTR}}$	$F_{\text{AFTER}}^{\text{DCTR}}$	$F_{\text{OVERLAP}}^{\text{DCTR}}$	$F_{\text{BEFORE/OVERLAP}}^{\text{DCTR}}$	$F_{\text{ALL}}^{\text{DCTR}}$	$F_{\text{CONTAINS}}^{\text{TLINK}}$	$F_{\text{BEFORE}}^{\text{TLINK}}$	$F_{\text{OVERLAP}}^{\text{TLINK}}$	$F_{\text{BEGINS-ON}}^{\text{TLINK}}$	$F_{\text{ENDS-ON}}^{\text{TLINK}}$	$F_{\text{ALL}}^{\text{TLINK}}$	
Baseline: perceptron (Khalifa et al., 2016)	-	-	-	-	0.528	0.759	0.456	0.147	0.073	0.060	0.024	0.364
(Lin et al., 2016b)	-	-	-	-	-	0.843	-	-	-	-	-	-
SP	0.837	0.805	0.860	0.575	0.833	0.608	0.294	0.185	0.158	0.231	0.518	
SP _{random sub-sampling}	0.837	0.803	0.859	0.575	0.833	0.564	0.275	0.204	0.154	0.218	0.490	
SP _{disjoint}	0.835	0.801	0.859	0.576	0.832	0.607	0.290	0.183	0.146	0.232	0.516	
SP ^{CC} + \mathcal{C}_*	0.843	0.810	0.861	0.573	0.836	0.603	0.292	0.186	0.148	0.222	0.514	
SP ^{UC} + \mathcal{C}_*	0.843	0.814	0.861	0.574	0.837	0.606	0.291	0.184	0.157	0.236	0.516	
SP + Φ_{odr}	0.856	0.830	0.867	0.569	0.846	0.608	0.291	0.182	0.159	0.222	0.518	
SP + Φ_{drtl}	0.838	0.811	0.855	0.564	0.831	0.605	0.286	0.176	0.147	0.217	0.514	

Table 5: Results on the THYME test set. SP refers to our structured perceptron model, without constraints or global features, using local initialization and loss-augmented negative sub-sampling. \mathcal{C}_* refers to using all constraints. Superscript CC and UC refer to using constraints at training and prediction time, or only at prediction time respectively.

5 Conclusions

In this work, we proposed a structured perceptron model for learning temporal relations between events and the document-creation time (DCTR), and between temporal entities in the text (TLINKS) in clinical records. Our model efficiently learns and predicts at a document level, exploiting loss-augmented negative sub-sampling, and uses global features allowing it to exploit relations between local output labels. For construction of a consistent output labeling, needed for time-line construction, we formulated a number of constraints, including those from (Chambers et al., 2007; Do et al., 2012), and assessed them during inference. Our best system outperforms the state-of-the-art of both the CONTAINS TLINK task, and the DCTR task. Our code for this work is available at <https://github.com/tuur/SPTempRels>.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments which helped us to improve the paper. Also, we would like to thank the Mayo Clinic for permission to use the THYME corpus. This work was funded by the KU Leuven C22/15/16 project "MACHINE READING OF PATIENT RECORDS (MARS)", and by the IWT-SBO 150056 project "ACQUIRING CRUCIAL MEDICAL INFORMATION USING LANGUAGE TECHNOLOGY" (AC-CUMULATE).

References

- Steven Bethard, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2014. Clinical tempeval. *arXiv preprint arXiv:1403.4928*.

Steven Bethard, Leon Derczynski, Guergana Savova, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. Semeval-2015 task 6: Clinical tempeval. In *Proceedings of SemEval*, pages 806–814. Association for Computational Linguistics.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. Semeval-2016 task 12: Clinical tempeval. pages 1052–1062. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP*, pages 698–706. Association for Computational Linguistics.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of ACL*, pages 173–176. Association for Computational Linguistics.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of ACL*, pages 489–496. Association for Computational Linguistics.

Carlo Combi and Yuval Shahar. 1997. Temporal reasoning and temporal data maintenance in medicine: issues and challenges. *Computers in Biology and Medicine*, 27(5):353–368.

Harold Charles Daume. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. ProQuest.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of EMNLP*, pages 677–687. Association for Computational Linguistics.

Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Inc. Gurobi Optimization. 2015. Gurobi optimizer reference manual.

- Abdulrahman Khalifa, Sumithra Velupillai, and Stephane Meystre. 2016. Utahbm1 at SemEval-2016 task 12: Extracting temporal information from clinical text. *Proceedings of SemEval*, pages 1256–1262.
- Parisa Kordjamshidi, Dan Roth, and Marie-Francine Moens. 2015. Structured learning for spatial information extraction from biomedical text: bacteria biotopes. *BMC Bioinformatics*, 16(1):1.
- Hee-Jin Lee, Yaoyun Zhang, Jun Xu, Sungrim Moon, Jingqi Wang, Yonghui Wu, and Hua Xu. 2016. UHealth at SemEval-2016 task 12: an end-to-end system for temporal information extraction from clinical notes. *Proceedings of SemEval*, pages 1292–1297.
- Artuur Leeuwenberg and Marie-Francine Moens. 2016. KULeuven-LIIR at SemEval 2016 task 12: Detecting narrative containment in clinical records. *Proceedings of SemEval*, pages 1280–1285.
- Chen Lin, Dmitriy Dligach, Timothy A Miller, Steven Bethard, and Guergana K Savova. 2016a. Multi-layered temporal modeling for the clinical domain. *Journal of the American Medical Informatics Association*, 23(2):387–395.
- Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2016b. Improving temporal relation extraction with training instance augmentation. *Proceedings of ACL*, page 108.
- Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of COLING–ACL*, pages 753–760. Association for Computational Linguistics.
- Luis Gerardo Mojica and Vincent Ng. 2016. Markov logic networks for text mining: A qualitative and empirical comparison with integer linear programming. In *Proceedings of LREC*, pages 4388–4395.
- Agnieszka Onisko, Allan Tucker, and Marek J. Druzdzel, 2015. *Prediction and Prognosis of Health and Disease*, pages 181–188. Springer International Publishing, Cham.
- James Pustejovsky and Amber Stubbs. 2011. Increasing informativeness in temporal annotation. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 152–160. Association for Computational Linguistics.
- James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The TimeBank corpus. In *Corpus Linguistics*, volume 2003, page 40.
- Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.
- Michael Stacey and Carolyn McGregor. 2007. Temporal abstraction in intelligent clinical data analysis: A survey. *Artificial Intelligence in Medicine*, 39(1):1–24.
- William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.
- Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813.
- Naushad UzZaman and James F Allen. 2011. Temporal evaluation. In *Proceedings of ACL–HLT*, pages 351–356. Association for Computational Linguistics.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*.
- Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with Markov logic. In *Proceedings of ACL-IJCNLP*, pages 405–413. Association for Computational Linguistics.

Entity Extraction in Biomedical Corpora: An Approach to Evaluate Word Embedding Features with PSO based Feature Selection

Shweta Yadav, Asif Ekbal, Sriparna Saha, Pushpak Bhattacharyya

Indian Institute of Technology Patna

Bihar, India

{shweta.pcs14, asif, sriparna, pb}@iitp.ac.in

Abstract

Text mining has drawn significant attention in recent past due to the rapid growth in biomedical and clinical records. Entity extraction is one of the fundamental components for biomedical text mining. In this paper, we propose a novel approach of feature selection for entity extraction that exploits the concept of deep learning and Particle Swarm Optimization (PSO). The system utilizes word embedding features along with several other features extracted by studying the properties of the datasets. We obtain an interesting observation that compact word embedding features as determined by PSO are more effective compared to the entire word embedding feature set for entity extraction. The proposed system is evaluated on three benchmark biomedical datasets such as GENIA, GENETAG and AiMed. The effectiveness of the proposed approach is evident with significant performance gains over the baseline models as well as the other existing systems. We observe improvements of 7.86%, 5.27% and 7.25% F-measure points over the baseline models for GENIA, GENETAG, and AiMed dataset respectively.

1 Introduction

The tremendous amount of information accumulated in the domains of molecular biology has drawn the attention of biomedical natural language processing (BioNLP) community in order to facilitate the development of various tools for various text processing applications, curation and organization of ever-growing biomedical literature etc. Entity extraction is crucial step for solving

several pipelined applications such as information extraction, automatic summarization, question-answering, word sense disambiguation etc. Biomedical entities mostly refer to the biological sequences of protein & gene such as DNA, RNA, cell_type, cell_line etc. (Kim et al., 2004). The way of extracting these information from biomedical and clinical texts refers to as entity extraction. An automatic system which can extract biomedical names such as gene, protein or any disease name from text can substantially reduce the human efforts. However, extracting these entities from text poses several challenges which are presented as follows:

1. Named entities are very generative in nature, i.e. many new names are continuously being generated. Any dictionary can not capture all the various forms of a given name.
2. Similar words convey different meanings, and therefore, a word can have multiple NE types. For example, gene names often contain alphabets, digits, hyphens, and other characters, thus having many variants (e.g., “HIV-1 enhancer” versus “HIV 1 enhancer”). Moreover, many abbreviations (e.g., “IL2” for “Interleukin 2”) constitute integral parts of biomedical named entities (NEs).
3. Biomedical names are usually of long length, and contains different types of symbols, and hence boundary detection becomes problematic.
4. Ambiguity: Same name could be used to represent variety of biological entities which further worsen the problem.

The challenges as of these kinds are the primary causes behind the low accuracies of the systems developed for entity extraction in biomedical

cal text. The research challenges have been addressed in the literature including in some shared-task challenges, such as JNLPBA (Joint Workshop on Natural Language Processing in Biomedicine and its Applications) in 2004 (Kim et al., 2004) and BioCreative (Critical Assessment for Information Extraction in Biology Challenge) II GM (gene mention) subtask in 2007 (Smith et al., 2008). Over the years several benchmark corpora have been created that do not conform to the uniform annotation guidelines. Therefore the system, developed by targeting a specific domain, often fails to show reasonable accuracy when it is evaluated for some other domains. In our work we attempt to build a system for entity extraction that performs well across various biomedical corpora.

Popular existing system mostly rely on rule-based system or supervised machine learning technique to automatically extract entities. They looked upon this problem as in terms of sequence labeling and used algorithm such as hidden markov models (HMM) (Zhao, 2004), support vector machines (SVM) (Kazama et al., 2002; GuoDong and Jian, 2004), maximum entropy Markov model (MEMM) (Finkel et al., 2005) and conditional random fields (CRF) (Ekbal et al., 2013; Settles, 2004; Kim et al., 2005). These supervised learning models is fully dependent on the features that we use for training. Some of the popular features used in the existing studies include linguistic features such as morphological, syntactic and semantic information of words and domain-specific features from biomedical ontologies such as Bio-Thesaurus (Liu et al., 2006) and UMLS (Unified Medical Language System) (Bodenreider, 2004). However, these features heavenly account to the problem of data sparsity.

In the recent past, there has been huge interest in using large unlabeled corpus to generate word representation feature using deep neural network technique. We are motivated by the strength of deep learning concepts to build our model. We use the well-known word embedding model that is a robust framework to incorporate word representation features (Mikolov et al., 2013b). Word representation feature is a mathematical description of the word in vector form. Each position of vector corresponds to a feature with some semantic or grammatical inference which leads to the term word feature. Word representation features contains latent syntactic/semantic informa-

tion of a word. The main objective to use word embedding is to provide more useful information to the model being trained. Vector based word representation has powerful capability that captures the phenomenon that words having the similar meanings should appear together (Mikolov et al., 2013b). In traditional machine learning, data sparsity is a problem that often causes the degradation in performance. This drawback could be overcome by the incorporation of word embedding with the presumption that similar type of word (as to semantics) appear in the similar context (Mikolov et al., 2013b).

The aim is to exploit the usefulness of neural network based word embedding (Bengio et al., 2003) as a feature for entity extraction in biomedical text. In addition we also make use of a very diverse feature set that exploits the properties of data and problem specific knowledge. We restrict ourselves from using much domain-specific information for feature extraction, keeping in view easy adaptability of the system to more than one biomedical corpora.

However, the huge dimensionality of the word representation vector often contributes to the complexity of the system. This motivated us to apply feature selection technique to reduce the dimensionality contributed by word embedding as well as to improve the system performance. Our algorithm for feature selection is based on wrapper based approach, which is formulated as an optimization problem. We use Particle Swarm Optimization(PSO) (Kennedy and Eberhart, 1997) as the underlying optimization strategy. Particle Swarm Optimization is an evolutionary technique, inspired by the social behavior of birds. Some recent studies show that PSO converges faster compared to some other widely used optimization techniques (Bansal et al., 2011). Inspired by this observation we use PSO in our current study.

To analyze the effect of pruned word embedding, we have carried out an experiment with all the handcrafted features and the reduced features as determined by PSO. We perform experiments on three standard datasets, namely GENIA, GENE-TAG and AiMed. Evaluation results show that we achieve significant performance gains with the use of pruned word embedding feature set. The best performance of the system was obtained when we apply PSO based feature selection technique on combination of handcrafted features set and word

embedding features. The key contribution of this paper are, (i) proposal of PSO based feature selection technique in bio-medical entity extraction. (ii) analysis of feature selection on only word representation features. (iii) impact of feature selection on word representation features with hand-craft features.

2 Related Works

There has been quite a significant number of existing works available for biomedical named entity recognition (BNER). These approaches can be divided into three major categories: (1) dictionary based, (2) rule based and (3) machine learning based techniques. Among these existing approaches, machine learning based techniques have gained a lot more attention due to the availability of sufficiently good amount of annotated corpus. For example, majority of the systems submitted to the JNLPBA challenge made use of machine learning algorithms which have been observed to significantly outperform the dictionary based methods.

Some of the recent works in BNER includes the unsupervised model as proposed in (Zhang and Elhadad, 2013), and the system based on CRF (Li et al., 2015a). A two-phase approach based on semi-Markov CRF is proposed in (Yang and Zhou, 2014). In the first phase boundaries of entities are identified while in the second phase semantic labeling is performed to label the detected entities. A CRF based system has been proposed by (Tang et al., 2015), where in the first step boundaries of NEs are identified and in the second step appropriate labels are assigned. (Grouin, 2014) performed experiments on the i2b2/VA-2010 challenge dataset to detect bacteria and biotopes names. They developed a model based on CRFs. An unsupervised approach is proposed in (Han et al., 2016) that made use of clustering based active learning. They have used Shared Nearest Neighbor (SNN) clustering technique. The work reported in (Li et al., 2015a), authors have proposed a parallel CRF algorithm (MapReduce CRF) which provides a mechanism to minimise the time taken for CRF learning. They showed that the proposed approach outperforms other traditional models in terms of time and efficiency. While, most of the proposed system used CRF, recently (Patra and Saha, 2013) proposed an entity extraction system based on SVM. Par-

ticularly, they have introduced a tree kernel based function that can efficiently solve the full NER task. The work proposed in (Tohidi et al., 2014) aims to improve the performance of entity extraction using statistical character-based syntax similarity (SCSS) algorithm. This algorithm computes the similarity between the identified candidate entities and a known set of well-known NEs. This set of NEs is created by extracting the most frequently occurring NEs in the GENIA V3.0 corpus. In recent times deep learning based approaches such as Recurrent Neural Network and Bi-directional LSTM have also used for entity extraction (Li et al., 2015b; Limsopatham and Collier, 2016). It is well known that relevant features play an important role for building a high accurate system. In our work, in addition to the standard features we also use the features extracted from the word embedding model.

Bengio et al. (Bengio et al., 2003) have proposed a neural network based model for vector representation of words. Distributed representation (also known as word embedding) of a word has been used to improve the performance of various NLP tasks like Part-of-Speech (POS) tagging, NER in news-wire domain (Collobert et al., 2011), parsing (Socher et al., 2013; Turian et al., 2010) etc. Word cluster has been used by Miller et al. (Miller et al., 2004) to boost the performance of a NER system. Tang et al. (Tang et al., 2012; Tang et al., 2013) have reported that performance of biomedical entity extraction can be improved when word representation is used as a feature to CRF and SVM classifiers.

Here we propose a PSO based feature selection technique that determines the most relevant features from a full word embedding set, and use this subset as feature for classifier's training. Feature selection has been widely used for many tasks such as gene expression (Ding and Peng, 2005), face recognition (Seal et al., 2015) and signal processing (Alamedine et al., 2013). Dealing with biomedical text is, however, more difficult and challenging as the features have non-numeric values and the texts are heavily unstructured. Except the few works such as NER (Ekbal and Saha, 2016), co-reference resolution (Sikdar et al., 2015) and sentiment analysis (Gupta et al., 2015), systematic methods of feature selection using meta-heuristics algorithms are very rare. Nevertheless, the importance of using pruned neural language

model based word representation features with effective feature selection have not been exploited so far in the literature.

2.1 A Brief Introduction to Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a meta-heuristic intelligent technique inspired by social behavior of the swarm for its survival (Eberhart and Shi, 1998; Kennedy and Eberhart, 1997). This is a population based technique which is perceived in birds and fishes for the search of the best path. In general, PSO consists of the swarm of the particle where each particle has its particular position in the search space with which it moves around the search space by some velocity. The particle selects the best path on each iteration by using its memory and by learning the effective path that was followed previously by the swarm. The new position is chosen on the basis of the knowledge gained previously by its self-best position and the best position of the swarm. PSO, being a meta-heuristic model, makes few or no assumption about the problem being optimized and can search very large spaces of candidate solutions. This makes PSO highly efficient for the optimization purpose (Yan et al., 2013). The algorithm iterates by keeping track of two variables: Global best position represents the most promising vector found so far, and Personal best position denotes the particle's own personal best solution.

2.1.1 Algorithm: PSO based Feature Selection

1. Initially, we randomly set the swarm population. Each particle of the swarm is represented by binary-valued features of length n (total no. of feature) and has its position and velocity with which it moves in search space. Mathematically, particle position and particle velocity are represented as: $\vec{P}(i)$ and $\vec{V}(i)$ respectively:

$$\vec{P}(i) = (p(i, 1), p(i, 2), \dots, p(i, n))$$

$$\vec{V}(i) = (v(i, 1), v(i, 2), \dots, v(i, n))$$

where $p(i, j) \in \{0, 1\}$, $i = 1, 2, \dots, N$ and $j = 1, 2, \dots, n$ where N is no. of particle. Particle maintains its best position ($\vec{B}(i)$) that they have achieved so far and also the global best position (\vec{G}) i.e., the best position of the particle having the best solution.

2. Particle's position $\vec{P}(i)$ value is set either $\{0, 1\}$ on the basis of following expression:

$$p_{(i,j)} = \begin{cases} 1 & \text{if } random \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

3. Each particle is evaluated on the basis of fitness function (F-measure value) $f(\vec{P}(i))$. The memory is updated by keeping track of the best position and global best position.
4. Initially, the value of best position ($\vec{B}(i)$) of every particle is set to 0. At every epoch(ep) the value of the best position is updated as follows:

$$f(\vec{B}(i))_{ep} = \max(f(\vec{P}(i))_{ep}, f(\vec{B}(i))_{ep-1})$$

5. Update in the global best position value is done when the fitness function $f(\vec{B}(i))$ in the swarm is superior than the existing $f(\vec{G})$.
6. Originally, the velocity vector is generated randomly. At each iteration, velocity of a particle is updated according to the following equation:

$$v_{(i,j)} = \omega * v_{(i,j)} + \phi_1(b_{(i,j)} - p_{(i,j)}) + \phi_2(g_{(j)} - p_{(i,j)}) \quad (1)$$

where ω ($0 < \omega < 1$), ϕ_1 and ϕ_2 are known as inertia weights. These parameters are initialized with an uniformly generated random numbers in the range (0,1). The $b_{(i,j)}$, $p_{(i,j)}$, and $g_{(j)}$ denote the j^{th} components of $\vec{B}(i)$, $\vec{P}(i)$ and \vec{G} , respectively.

7. The position of a particle is updated by the following mathematical expression:

$$p_{(i,j)} = \begin{cases} 1 & \text{if } (random < S(v_{(i,j)})) \\ 0 & \text{otherwise} \end{cases}$$

where $0 \leq random \leq 1$ is an uniform random number.

$$S(v_{(i,j)}) = \frac{1}{1 + \exp(-v_{(i,j)})}$$

This represents the sigmoid function. Thus, we update the particle position value of 0 or 1 on the basis of the value of velocity.

8. Repeat steps 4-7 until convergence.

2.2 Learning Word Representations

Word embedding (also known as distributed word representations) persuade a real-valued latent semantic or syntactic vector for each word from a large unlabeled corpus by using continuous space language models (Tang et al., 2014). Better word representation can be obtained if we have a large amount of training data as the obtained real-valued vectors of words become more representative. We use the popular *word2vec*¹ tool proposed by Mikolov et al. (Mikolov et al., 2013a) to extract the vector representations of words. Owing to its simpler architecture which reduces the computational complexity, this technique can be used for large corpus. Two models have been proposed in (Mikolov et al., 2013a) to learn vector representation known as Continuous Bag-of-Words Model (CBOW) and Skip-gram model. Since skip-gram model is able to capture the semantic information of a word, we adapt this to train the model for vector representation. The Skip-gram architecture tries to maximize the classification of a word based on the other words in the same sentence. More formally, given a sequence of training words w_1, w_2, \dots, w_T , the objective of the Skip-gram model is to maximize the average log probability

$$\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \log p(w_{t+j}|w_t) \quad (2)$$

where c is the window size. Here, we show few words that are more nearby to any biomedical entity: ‘antigen’, ‘lymphocytes’ and ‘inhibited’. If we look at the most similar words for the word ‘lymphocytes’, we observe that apart from syntactically similar words like ‘T-lymphocytes’, ‘B-lymphocytes’, it is also able to capture the words which are semantically similar like ‘CD3+’, ‘PBLs’ and ‘T-cells’.

3 Features for Entity Extraction

The features being extracted are described as follows:

1. Contextual feature: It is the local contextual feature which refers to the tokens which appear within the window size of 10 words, i.e 5 to the left and 5 to the right w.r.t current token.

¹<https://code.google.com/p/word2vec/>

2. Word prefixes and suffixes: These features refer to the fixed length character sequences stripped either from the left or rightmost positions of the words.
3. Word length: It is observed that short words are rarely the NEs. We define a binary-valued feature that triggers the value 1 if the length of current word is greater than the threshold value specified. The threshold value is set as 5 in this case.
4. Part-of-Speech (PoS) information: PoS provides useful syntactic evidence for detecting named entities (NEs). We use PoS information of the current and/or the surrounding token(s) as the feature. The PoS information was extracted from the GENIA tagger² V2.0.2.
5. Chunk information: We use GENIA tagger V2.0.2 corpus to extract the chunk information. We employ the chunk information of the present and neighboring tokens as the features.
6. Word shape: Word shape is defined as the mapping of each word to its equivalent class. In order to implement this feature we normalize the words by converting every capital character by ‘A’, small character to ‘a’ and digit to ‘0’. After this conversion, we squeeze the consecutive characters into a single character. For example, if we consider the token ‘Ly-49’, the normalized word for this token would be ‘Aa-00’.
7. Word class feature : This feature is based on the concept that entities present in the same class are mostly similar. Here, all the capital letters are converted to ‘A’, small letters to ‘a’, numbers to ‘O’ and non-English characters to ‘-’. After this conversion, we squeeze the consecutive characters into a single character. For example, the word class feature for the token ‘IL-2-mediated’ is ‘AA-O-aaaaaaaa’, which is further reduced to ‘A-O-a’.
8. Orthographic features: We use several orthographic features that consider capitalization and digit information. These features are:

²<http://www.nactem.ac.uk/GENIA/tagger/>

initial capital, all capital, capital in inner, initial capital then mix, only digits, digit with special character, initial digit then alphabet, digit in inner. It is observed that some symbols like (‘,’ ‘-’, ‘.’, ‘_’) are very common in the biomedical text. Some symbols like ‘;’ are also very helpful for the identification of NE boundaries.

4 Methodology

We propose a PSO based feature selection technique that determines the most relevant features from a set of features, containing both handcrafted as well as word embedding based features. We use Conditional Random Field (CRF) (Lafferty et al., 2001) as a base learning algorithm. For each token, a feature vector is generated from the training and test dataset using the features as described in the previous section. Basic steps of our algorithm are as follows:

1. Initially, we design 32 features (listed in Section-3) for three datasets, namely GENIA, GENETAG and AiMed. These features are used for the classifier’s training. The models built using these features are termed as the baseline models.
2. We generate the word embedding feature vector of 200 dimensions based on the model trained on a large corpus like Wikipedia and the biomedical corpora such as PubMed³ and PubMed Central Open Access (PMC OA)⁴.
3. A new feature set is generated by combining both word embedding based features and handcrafted features.
4. PSO based feature selection is performed to determine the most relevant feature set.
5. CRF classifier is trained with the features selected by PSO. The model, thus generated, is evaluated on all the three datasets.

Figure-1 depicts the various steps of our proposed approach.

5 Datasets and Experiments

³<http://www.ncbi.nlm.nih.gov/pubmed>

⁴<http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist>

5.1 Dataset

Our system is evaluated on three distinguished biomedical datasets, namely GENIA⁵, AiMed⁶ and GENETAG⁷. The GENIA corpus is derived from the MEDLINE corpus. It comprises of 500,000 and 100,000 words in training and test dataset, respectively. These datasets are manually annotated with five NE tags, namely *Protein*, *DNA*, *RNA*, *Cell_line* & *Cell_type*.

AiMed corpus was created using 20,000 sentences having gene/protein names extracted from the Database of Interacting Protein (DIP). We use 7,500 labeled sentences for training and 2,500 sentences for validation. For evaluation we use a test set consisting of 5,000 sentences.

GENETAG dataset is derived from the ‘Med-Tag’ dataset. Training and test datasets comprise of 118K and 142K words, respectively. In order to properly denote the boundaries of NE, we use the IOB2⁸ encoding scheme. We evaluate our system in terms of recall, precision and F-measure values. For evaluation we use the script, which was made available with the JNLPBA 2004 shared task⁹.

5.2 Baseline Models and Analysis

We start experiments with the first baseline (i.e. Baseline-1) by developing the model trained with all the features as discussed in Section-3. We evaluate the presence of word embedding features trained on various unlabeled data sets obtained from the different text sources. In order to realize the effect of each trained word representation model, we augment the word vector obtained from the respective model one by one to the baseline feature set. In order to obtain word embedding, we use four different models trained on the unlabeled data extracted from PubMed¹⁰, PubMed Central Open Access (PMC OA)¹¹ and the latest English Wikipedia dump¹². Corpus statistics of PubMed and PMC OA are provided in Table-1. The extracted text upon which four different models are trained are as follows:

⁵<http://www.geniaproject.org/shared-tasks/bionlp-jnlpba-shared-task-2004>

⁶<ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/>

⁷<ftp://ftp.ncbi.nlm.nih.gov/pub/tanabe/GENETAG.tar.gz>

⁸I, O and B represent the intermediate, outside and beginning token of a NE

⁹<http://www.nactem.ac.uk/tsujii/GENIA/ERTask/report.html>

¹⁰<http://www.ncbi.nlm.nih.gov/pubmed>

¹¹<http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist>

¹²http://en.wikipedia.org/wiki/Main_Page

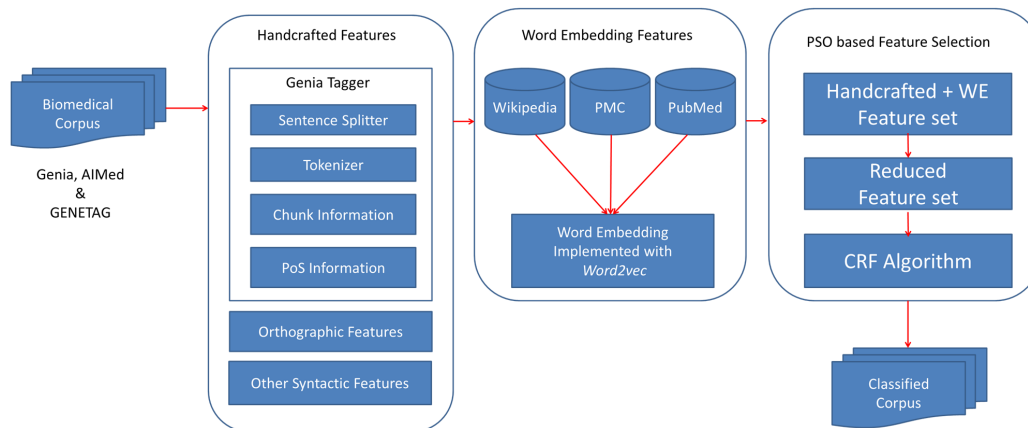


Figure 1: Proposed model architecture for biomedical entity extraction

1. Model developed using extracted data from PubMed biomedical literature: denoted as WE(1).
2. Model built on the extracted text from PMC biomedical literature: denoted as WE(2).
3. Model developed using the combination of extracted text from PubMed and PMC biomedical texts: denoted as WE(3).
4. Model trained using the extracted text from PubMed, PMC and Wikipedia: denoted as WE(4).

We develop the second baseline (i.e. Baseline-2) by executing the best word embedding model in combination with the hand-crafted feature set. We further develop the third baseline, i.e. Baseline-3 by merging word embedding feature set as determined by PSO along with the full handcrafted feature set. We observe that selection of relevant word embedding features helps in improving performance over the whole word embedding feature set.

We generate 200-dimensional word vectors using the parameters¹³ as follows: *skip-gram* model with a window size of 5, hierarchical soft-max training, and a frequent word sub-sampling threshold of 0.001. In order to make our proposed system generic, i.e. not biased to any particular domain of data, we use the same parameters of PSO in all our settings. We fine-tune the parameters ω , ϕ_1 and ϕ_2 by performing 3-fold cross validation experiments. We keep the number of particles

¹³We use same parameters for training of all the four models

Corpus	Documents	Sentences	Tokens
PubMed	22,120,269	124,615,674	2,896,348,481
PMC	672,589	105,194,341	2,591,137,744
PubMed+PMC	22,792,858	229,810,015	5,487,486,225

Table 1: Corpus statistics (Pyysalo et al., 2013) of PubMed and PMC OA openly available biomedical literature; PubMed abstracts for articles that are also present in PMC OA were discarded while creating the data

and the number of iterations as 10 and 100, respectively throughout all the experiments.

Effectiveness of PSO based feature selection is evident with performance improvement as shown in Table-5.

5.3 Comparison with Existing Feature Selection Techniques

Here we compare our PSO based feature selection technique with other existing feature selection techniques. We perform experiments with both filter and wrapper based models. For filter based model, we use univariate feature selection based on information theoretical concept like Information Gain. While for multivariate filter model we use correlation based feature selection. Our results indicate that PSO performs better than univariate by 3.03 % and multivariate by 2.60 % F-measure points for the GENIA dataset. We also observe quite similar behaviors for the other two datasets.

In addition, we also explore two popular wrapper based feature selection techniques, Genetic Algorithm (GA) (Holland, 1975) based feature selection (Ekbal et al., 2010) technique and Recursive Feature Elimination (RFE) (Guyon et al., 2002)

Feature Selection	GENIA			GENETAG			AiMed		
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
Filter (Information Gain)	69.02	71.28	70.13	88.25	94.47	91.25	87.46	90.47	88.93
Filter (Correlation)	70.19	70.95	70.56	88.89	93.68	91.22	87.18	89.85	88.49
Wrapper (GA)	72.48	71.98	72.22	89.19	95.04	92.02	89.07	91.11	90.07
Wrapper (RFE)	71.28	71.54	71.40	89.25	94.81	91.94	88.67	91.66	90.14
CRF[PSO]	72.48	73.87	73.16	89.33	96.42	92.74	89.77	92.09	90.92

Table 2: Comparison of PSO with other filter (Information Gain & Correlation) and wrapper (G.A & RFE) based feature selection technique

based approach. Genetic algorithm belongs to the class of randomized wrapper model where feature selection is always classifier dependent and is less prone to stuck at local optima. The RFE is categorized under the deterministic type wrapper model which is computationally less complex than randomized type but has the disadvantage to stuck at local optima.

Results show that PSO performs better than RFE for all the datasets and GA for two datasets (GENIA & GENETAG) in terms of F-measure and the number of features selected. Results are depicted in Table-2. On AiMed dataset, GA and RFE based feature selection techniques perform quite comparable to our PSO based method. It is to be noted that PSO based feature selection yields better performance even with a smaller set of features. The pruned and compact feature set incurs less computational complexity.

6 Result and Discussion

Table-3 shows the extensive results of our proposed system on all three datasets, namely GENIA, AiMed and GENETAG by augmenting word embedding features. It seems that word embedding features generated from the model which is trained on the combined datasets of PubMed, PMC and Wikipedia [WE(4)] perform better than the other models. The unsupervised word representation features help in detecting unseen entities, i.e. those not appearing in the training data set.

We augment word embedding WE(4) features to the hand-crafted features, and then apply feature selection using PSO on this combined set. Feature selection through PSO not only helps in improving the performance, but at the same time it reduces the feature dimensionality. Evaluation results as reported in Table-4 reveal this fact. Table-3 clearly depicts the effectiveness

of word embedding features in BNER (biomedical NER) system. We observe improvements of 7.86%, 5.27% and 7.25% F-measures over the first baseline (i.e. Baseline-1) for GENIA, AiMed and GENETAG data sets, respectively by using PSO based feature selection on PubMed-PMC-wikipedia trained word embedding and handcrafted features. Evaluation also suggests that performance does not degrade significantly, even when we use word embedding features obtained only from Pubmed & PMC OA. It seems that word embedding features obtained from the combination of Pubmed and PMC are more representatives compared to the individual one.

We also show evaluation of some of the existing approaches that attempt to make use of word representation features. A F-measure of 71.39% is reported in the work (Tang et al., 2014). Word representation feature was also used in (Chang et al., 2015) that reported to have achieved F-measure value of 71.77%.

We perform statistical significance (t-test) test on the results obtained by our proposed model. For different datasets, experiments are executed for 10 independent runs and the t-statistic is adopted to analyze the obtained experimental results. Using the known distribution of the test statistic, p -value is calculated. It is observed that p values are less than 0.04 for all the three data sets, which signify that our obtained results are statistically significant.

6.1 Error Analysis

Here, we analyze the outputs obtained for each dataset in order to identify the possible errors. We categorize the errors in three ways as follows:

1. Wrong boundary: This error occurs due to the incorrect boundary identification of entities. These types of cases are observed

System	GENIA			AiMed			GENETAG		
	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
Baseline-1: Handcraft feature	66.78	63.89	65.30	84.11	87.25	85.65	83.88	87.17	85.49
WE1(Handcraft feature + PubMed)	70.72	72.29	71.50	88.42	89.21	88.81	81.92	95.82	88.33
WE2(Handcraft feature + PMC OA)	70.72	72.29	71.50	88.56	89.02	88.78	82.01	95.62	88.29
WE3(Handcraft feature + PubMed + PMC)	70.79	72.47	71.62	89.48	89.01	89.24	82.41	95.89	88.64
WE4(Handcraft feature + PubMed + PMC + Wikipedia)	70.88	72.64	71.75	89.07	90.11	89.59	82.78	95.70	88.77
Baseline-2: Best of WE model	70.88	72.64	71.75	89.07	90.11	89.59	82.78	95.70	88.77
Baseline-3: (PSO with only WE(4)) + Handcraft feature	71.92	72.62	72.26	89.63	90.34	89.98	83.46	95.69	89.15
Proposed: PSO with (handcrafted features + WE)	72.48	73.87	73.16	89.77	92.09	90.92	89.33	96.42	92.74
WE model by Tang et al.(Tang et al., 2014))	70.78	72.00	71.39	-	-	-	-	-	-
WE model by Chang et al.(Chang et al., 2015))	71.36	72.18	71.77	-	-	-	-	-	-

Table 3: Performance evaluation on GENIA, AiMed and GENETAG data sets using various word embedding (WE) features trained on different unlabeled data.

Approach	Dataset		
	GENIA	GENETAG	AiMed
Handcraft Features + W.E Features	232	230	232
PSO based feature selection	129	136	121

Table 4: Comparison of no. of features being used to train the model: Before feature selection and after feature selection

Approach	Dataset		
	GENIA	GENETAG	AiMed
Only W.E features	57.78	55.63	41.22
PSO selected W.E features	58.96	57.41	42.74

Table 5: Comparisons (in terms of F-score) between whole word embedding features using WE(4) and the PSO selected word embedding features excluding handcrafted features. Here, W.E: Word embedding

mostly with the entities having long and compounded wordforms such as ‘T cell activation-specific enhance’. We also observe that our system lacks in correctly classifying the instances which includes brackets.

2. Incorrect entity type: This error is obtained when the entity is properly identified but it belongs to some other entity class. This error is more prominent in case of GENIA and GENETAG datasets. For GENIA dataset, classifier is mostly confused with ‘Protein’ vs. ‘Cell_line’ or ‘Cell_type’. In total 126 Protein words are wrongly classified either as the ‘Cell_line’ or ‘Cell_type’. While with the use of PSO, the rate of mis-classification was reduced to 97. In GENETAG, majority of classes are predicted as ‘I-NEWGENE’. This may be due to the fact that majority of the instances belongs to the ‘I-NEWGENE’ cat-

egory. While after applying PSO, we observe that mis-classification of ‘I-NEWGENE’ is significantly reduced from 325 to just 129.

3. Missed entity: Our system misses significant number of NE instances. It is found that number of false negatives count to 1357, 155 and 40 for GENIA, AiMed and GENETAG, respectively. All these NEs are mis-classified to belong to the other-than-NE category.

7 Conclusions & Future work

In this paper we have investigated the effect of word embedding features in addition to the handcrafted features for entity extraction from three benchmark biomedical data sets, namely GENIA, AiMed & GENETAG. We have evaluated the system using four different word representation schemes trained on extracted texts from PubMed, PMC OA biomedical literature and Wikipedia dump datasets. In addition to this we have performed PSO based feature selection on the whole feature set for the different data sets. We can conclude that instead of using a full word representation feature, if only prominent features are used, it could help in improving the performance of the system. In future work, we would like to perform additional experiments to fine-tune the dimensions of vectors and the parameters of CRF through cross-validation on the training set. The applicability of feature selection on word embedding features need to be explored in other domain also. In addition we want to compare the performance of representation obtained through *word2vec* to the others such as GloVe. We would also like to explore deep learning techniques replacing CRF.

References

- Dima Alamedine, Catherine Marque, and Mohamad Khalil. 2013. Binary particle swarm optimization for feature selection on uterine electrohystero-gram signal. In *Advances in Biomedical Engineering (ICABME), 2013 2nd International Conference on*, pages 125–128. IEEE.
- J.C. Bansal, P.K. Singh, Mukesh Saraswat, Abhishek Verma, Shimpi Singh Jadon, and Ajith Abraham. 2011. Inertia weight strategies in particle swarm optimization. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pages 633–640. IEEE.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.
- Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270.
- F.X. Chang, J. Guo, W.R. Xu, and S. Rely Chung. 2015. Application of word embeddings in biomedical named entity recognition tasks. *Journal of Digital Information Management*, 13(5).
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.
- Chris Ding and Hanchuan Peng. 2005. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(02):185–205.
- Russell C. Eberhart and Yuhui Shi. 1998. Comparison between genetic algorithms and particle swarm optimization. In *Evolutionary Programming VII*, pages 611–616. Springer.
- Asif Ekbal and Sriparna Saha. 2016. Simultaneous feature and parameter selection using multiobjective optimization: application to named entity recognition. *Int. J. Machine Learning & Cybernetics*, 7(4):597–611.
- Asif Ekbal, Sriparna Saha, Utpal Kumar Sikdar, and Md Hasanuzzaman. 2010. A genetic approach for biomedical named entity recognition. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 354–355. IEEE.
- Asif Ekbal, Sriparna Saha, and Utpal Kumar Sikdar. 2013. Biomedical named entity extraction: some issues of corpus compatibilities. *Springer-Plus*, 2(1):1–12.
- Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: gene and protein identification in biomedical text. *BMC bioinformatics*, 6(1):1.
- Cyril Grouin. 2014. Biomedical entity extraction using machine-learning based approaches. *substance*, 6:1–611.
- Zhou GuoDong and Su Jian. 2004. Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 96–99. Association for Computational Linguistics.
- Deepak Kumar Gupta, Kandula Srikanth Reddy, Shweta, and Asif Ekbal. 2015. Pso-asent: Feature selection using particle swarm optimization for aspect based sentiment analysis. In *Natural Language Processing and Information Systems - 20th International Conference on Applications of Natural Language to Information Systems, NLDB 2015 Passau, Germany, June 17-19, 2015 Proceedings*, pages 220–233.
- Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422.
- Xu Han, Chee Keong Kwoh, and Jung-jae Kim. 2016. Clustering based active learning for biomedical named entity recognition. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1253–1260. IEEE.
- John H. Holland. 1975. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.
- Jun’ichi Kazama, Takaki Makino, Yoshihiro Ohta, and Jun’ichi Tsujii. 2002. Tuning support vector machines for biomedical named entity recognition. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain-Volume 3*, pages 1–8. Association for Computational Linguistics.
- James Kennedy and Russell C. Eberhart. 1997. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 5, pages 4104–4108. IEEE.
- Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics.

- Seonho Kim, Juntae Yoon, Kyung-Mi Park, and Hae-Chang Rim. 2005. Two-phase biomedical named entity recognition using a hybrid method. In *Natural Language Processing–IJCNLP 2005*, pages 646–657. Springer.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289.
- Kenli Li, Wei Ai, Zhuo Tang, Fan Zhang, Lingang Jiang, Keqin Li, and Kai Hwang. 2015a. Hadoop recognition of biomedical named entity using conditional random fields. *IEEE Transactions on Parallel and Distributed Systems*, 26(11):3040–3051.
- Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. 2015b. Biomedical named entity recognition based on extended recurrent neural networks. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 649–652. IEEE.
- Nut Limsopatham and Nigel Collier. 2016. Learning orthographic features in bi-directional LSTM for biomedical named entity recognition. *BioTxtM 2016*, page 10.
- Hongfang Liu, Zhang-Zhi Hu, Jian Zhang, and Cathy Wu. 2006. Biothesaurus: a web-based thesaurus of protein and gene names. *Bioinformatics*, 22(1):103–105.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 337–342, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.
- Rakesh Patra and Sujana Kumar Saha. 2013. A kernel-based approach for biomedical named entity recognition. *The Scientific World Journal*, 2013.
- S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou. 2013. Distributional semantics resources for biomedical text processing. In *Proceedings of LBM 2013*, pages 39–44.
- Ayan Seal, Suranjan Ganguly, Debotosh Bhattacharjee, Mita Nasipuri, and Consuelo Gonzalo-Martin. 2015. Feature selection using particle swarm optimization for thermal face recognition. In *Applied Computation and Security Systems*, pages 25–35. Springer.
- Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics.
- Utpal Kumar Sikdar, Asif Ekbal, Sriparna Saha, Olga Uryupina, and Massimo Poesio. 2015. Differential evolution-based feature selection technique for anaphora resolution. *Soft Comput.*, 19(8):2149–2161.
- Larry Smith, Lorraine K. Tanabe, Rie J. Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M. Friedrich, Kuzman Ganchev, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(Suppl 2):S2.
- Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. 2012. Clinical entity recognition using structural support vector machines with rich features. In *Proceedings of the ACM sixth international workshop on Data and text mining in biomedical informatics*, pages 13–20. ACM.
- Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. 2013. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. *BMC medical informatics and decision making*, 13(Suppl 1):S1.
- Buzhou Tang, Hongxin Cao, Xiaolong Wang, Qingcai Chen, and Hua Xu. 2014. Evaluating word representation features in biomedical named entity recognition tasks. *BioMed research international*, 2014.
- Zhuo Tang, Lingang Jiang, Li Yang, Kenli Li, and Keqin Li. 2015. Crfs based parallel biomedical named entity recognition algorithm employing mapreduce framework. *Cluster Computing*, 18(2):493–505.
- Hossein Tohidi, Hamidah Ibrahim, and Masrah Azrifah Azmi Murad. 2014. Improving named entity recognition accuracy for gene and protein in biomedical text literature. *International journal of data mining and bioinformatics*, 10(3):239–268.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

- Xuesong Yan, Qinghua Wu, Hanmin Liu, and Wenzhi Huang. 2013. An improved particle swarm optimization algorithm and its application. *International Journal of Computer Science Issues (IJCSI)*, 10(1).
- Li Yang and Yanhong Zhou. 2014. Exploring feature sets for two-phase biomedical named entity recognition using semi-crfs. *Knowledge and information systems*, 40(2):439–453.
- Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.
- Shaojun Zhao. 2004. Named entity recognition in biomedical texts using an hmm model. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 84–87. Association for Computational Linguistics.

Distant Supervision for Relation Extraction beyond the Sentence Boundary

Chris Quirk and Hoifung Poon

Microsoft Research

One Microsoft Way

Redmond, WA 98052

{chrisq, hoifung}@microsoft.com

Abstract

The growing demand for structured knowledge has led to great interest in relation extraction, especially in cases with limited supervision. However, existing distance supervision approaches only extract relations expressed in single sentences. In general, cross-sentence relation extraction is under-explored, even in the supervised-learning setting. In this paper, we propose the first approach for applying distant supervision to cross-sentence relation extraction. At the core of our approach is a graph representation that can incorporate both standard dependencies and discourse relations, thus providing a unifying way to model relations within and across sentences. We extract features from multiple paths in this graph, increasing accuracy and robustness when confronted with linguistic variation and analysis error. Experiments on an important extraction task for precision medicine show that our approach can learn an accurate cross-sentence extractor, using only a small existing knowledge base and unlabeled text from biomedical research articles. Compared to the existing distant supervision paradigm, our approach extracted twice as many relations at similar precision, thus demonstrating the prevalence of cross-sentence relations and the promise of our approach.

1 Introduction

The accelerating pace in technological advance and scientific discovery has led to an explosive growth in knowledge. The ensuing information overload creates new urgency in assimilating frag-

mented knowledge for integration and reasoning. A salient case in point is precision medicine (Bahcall, 2015). The cost of sequencing a person's genome has fallen below \$1000¹, enabling individualized diagnosis and treatment of complex genetic diseases such as cancer. The availability of measurement for 20,000 human genes makes it imperative to integrate all knowledge about them, which grows rapidly and is scattered in millions of articles in PubMed². Traditional extraction approaches require annotated examples, which makes it difficult to scale to the explosion of extraction demands. Consequently, there has been increasing interest in indirect supervision (Banko et al., 2007; Poon and Domingos, 2009; Toutanova et al., 2015), with distant supervision (Craven et al., 1998; Mintz et al., 2009) emerging as a particularly promising paradigm for augmenting existing knowledge bases from unlabeled text (Poon et al., 2015; Parikh et al., 2015).

This progress is exciting, but distant-supervision approaches have so far been limited to single sentences, thus missing out on relations crossing the sentence boundary. Consider the following example: “*The p56Lck inhibitor **Dasatinib** was shown to enhance apoptosis induction by dexamethasone in otherwise GC-resistant CLL cells. This finding concurs with the observation by Sade showing that **Notch**-mediated resistance of a mouse lymphoma cell line could be overcome by inhibiting p56Lck.*” Together, the two sentences convey the fact that the drug *Dasatinib* could overcome resistance conferred by mutations to the *Notch* gene, which can not be inferred from either sentence alone. The impact of missed opportunities is especially pronounced in the long tail of knowledge. Such information is crucial for integrative reasoning as it includes the newest

¹<http://www.illumina.com/systems/hiseq-x-sequencing-system.html>

²<http://www.ncbi.nlm.nih.gov/pubmed>

findings in specialized domains.

In this paper, we present *DISCREX*, the first approach for distant supervision to relation extraction beyond the sentence boundary. The key idea is to adopt a document-level graph representation that augments conventional intra-sentential dependencies with new dependencies introduced for adjacent sentences and discourse relations. It provides a unifying way to derive features for classifying relations between entity pairs. As we augment this graph with new arcs, the number of possible paths between entities grow. We demonstrate that feature extraction along multiple paths leads to more robust extraction, allowing the learner to find structural patterns even when the language varies or the parser makes an error.

The cross-sentence scenario presents a new challenge in candidate selection. This motivates our concept of *minimal-span candidates* in Section 3.2. Excluding non-minimal candidates substantially improves classification accuracy.

There is a long line of research on discourse phenomena, including coreference (Haghighi and Klein, 2007; Poon and Domingos, 2008; Rahman and Ng, 2009; Raghunathan et al., 2010), narrative structures (Chambers and Jurafsky, 2009; Cheung et al., 2013), and rhetorical relations (Marcu, 2000). For the most part, this work has not been connected to relation extraction. Our proposed extraction framework makes it easy to integrate such discourse relations. Our experiments evaluated the impact of coreference and discourse parsing, a preliminary step toward in-depth integration with discourse research.

We conducted experiments on extracting drug-gene interactions from biomedical literature, an important task for precision medicine. By bootstrapping from a recently curated knowledge base (KB) with about 162 known interactions, our *DISCREX* system learned to extract inter-sentence drug-gene interactions at high precision. Cross-sentence extraction doubled the yield compared to single-sentence extraction. Overall, by applying distant supervision, we extracted about 64,000 distinct interactions from about one million PubMed Central full-text articles, attaining two orders of magnitude increase compared to the original KB.

2 Related Work

To the best of our knowledge, distant supervision has not been applied to cross-sentence relation ex-

traction in the past. For example, Mintz et al. (2009), who coined the term “distant supervision”, aggregated features from multiple instances for the same relation triple (relation, entity1, entity2), but each instance is a sentence where the two entities co-occur. Thus their approach cannot extract relations where the two entities reside in different sentences. Similarly, Zheng et al. (2016) aggregated information from multiple sentential instances, but could not extract cross-sentence relations.

Distant supervision has also been applied to completing Wikipedia Infoboxes (Wu and Weld, 2007) or TAC KBP Slot Filling³, where the goal is to extract attributes for a given entity, which could be considered a special kind of relation triples (attribute, entity, value). These scenarios are very different from general cross-sentence relation extraction. For example, the entity in consideration is often the protagonist in the document (title entity of the article). Moreover, state-of-the-art methods typically consider extracting from single sentences only (Surdeanu et al., 2012; Surdeanu and Ji, 2014; Koch et al., 2014).

In general, cross-sentence relation extraction has received little attention, even in the supervised-learning setting. Among the limited amount of prior work, Swampillai & Stevenson (2011) is the most relevant to our approach, as it also considered syntactic features and introduced a dependency link between the root nodes of parse trees containing the given pair of entities. However, the differences are substantial. First and foremost, their approach used standard supervised learning rather than distant supervision. Moreover, we introduced the document-level graph representation, which is much more general, capable of incorporating a diverse set of discourse relations and enabling the use of rich syntactic and surface features (Section 3). Finally, Swampillai & Stevenson (2011) evaluated on MUC6⁴, which contains only 318 Wall Street Journal articles. In contrast, we evaluated on large-scale extraction from about one million full-text articles and demonstrated the large impact of cross-sentence extraction for an important real-world application.

The lack of prior work in cross-sentence relation extraction may be partially explained by the domains of focus. Prior extraction work focuses

³<http://www.nist.gov/tac/2016/KBP/ColdStart/index.html>

⁴<https://catalog.ldc.upenn.edu/LDC2003T13>

on newswire text⁵ and the Web (Craven et al., 2000). In these domains, the extracted relations often involve popular entities, for which there often exist single sentences expressing the relation (Banko et al., 2007). However, there is much less redundancy in specialized domains such as the frontiers of science and technology, where cross-sentence extraction is more likely to have a significant impact. The long-tailed characteristics of such domains also make distant supervision a natural choice for scaling up learning. This paper represents a first step toward exploring the confluence of these two directions.

Distant supervision has been extended to capture implicit reasoning, via matrix factorization or knowledge base embedding (Riedel et al., 2013; Toutanova et al., 2015; Toutanova et al., 2016). Additionally, various models have been proposed to address the noise in distant supervision labels (Hoffmann et al., 2011; Surdeanu et al., 2012). These directions are orthogonal to cross-sentence extraction, and incorporating them will be interesting future work.

Recently, there has been increasing interest in relation extraction for biomedical applications (Kim et al., 2009; Nédellec et al., 2013). However, past methods are generally limited to single sentences, whether using supervised learning (Björne et al., 2009; Poon and Vanderwende, 2010; Riedel and McCallum, 2011) or distant supervision (Poon et al., 2015; Parikh et al., 2015).

The idea of leveraging graph representations has been explored in many other settings, such as knowledge base completion (Lao et al., 2011; Gardner and Mitchell, 2015), frame-semantic parsing (Das and Smith, 2011), and other NLP tasks (Radev and Mihalcea, 2008; Subramanya et al., 2010). Linear and dependency paths are popular features for relation extraction (Snow et al., 2006; Mintz et al., 2009). However, past extraction focuses on single sentences, and typically considers the shortest path only. In contrast, we allow interleaving edges from dependency and word adjacency, and consider top K paths rather than just the shortest one. This resulted in substantial accuracy gain (Section 4.5).

There has been prior work on leveraging coreference in relation extraction, often in the standard supervised setting (Hajishirzi et al., 2013; Durrett

and Klein, 2014), but also in distant supervision (Koch et al., 2014; Augenstein et al., 2016). Notably, while Koch et al. (2014) and Augenstein et al. (2016) still learned to extract from single sentences, they augmented mentions with coreferent expressions to include linked entities that might be in a different sentence. We explored the potential of this approach in our experiments, but found that it had little impact in our domain, as it produced few additional candidates beyond single sentences. Recently, discourse parsing has received renewed interest (Ji and Eisenstein, 2014; Feng and Hirst, 2014; Surdeanu et al., 2015), and discourse information has been shown to improve performance in applications such as question answering (Sharp et al., 2015). In this paper, we generated coreference relations using the state-of-the-art Stanford coreference systems (Lee et al., 2011; Recasens et al., 2013; Clark and Manning, 2015), and generated rhetorical relations using the winning approach (Wang and Lan, 2015) in the CoNLL-2015 Shared Task on Discourse Parsing.

3 Distant Supervision for Cross-Sentence Relation Extraction

In this section, we present DISCREX, short for *Distant Supervision for Cross-sentence Relation EXtraction*. Similar to conventional approaches, DISCREX learns a classifier to predict the relation between two entities, given text spans where the entities co-occur. Unlike most existing methods, however, DISCREX allows text spans comprising multiple sentences and explores potentially many paths between these entities.

3.1 Distant Supervision

Like prior approaches, DISCREX learns from an existing knowledge base (KB) and unlabeled text. The KB contains known instances for the given relation. In a preprocessing step, relevant entities are annotated within this text using available entity extraction tools. Co-occurring entity pairs known to have the relation in the KB are chosen as positive examples. Under the assumption that related entities are relatively rare, we randomly sample co-occurring entity pairs not known to have the relation as negative examples. To ensure a balanced training set, we always sampled roughly the same number of negative examples as positive ones.

⁵E.g., MUC6, ACE <https://www ldc.upenn.edu/collaborations/past-projects/ace>

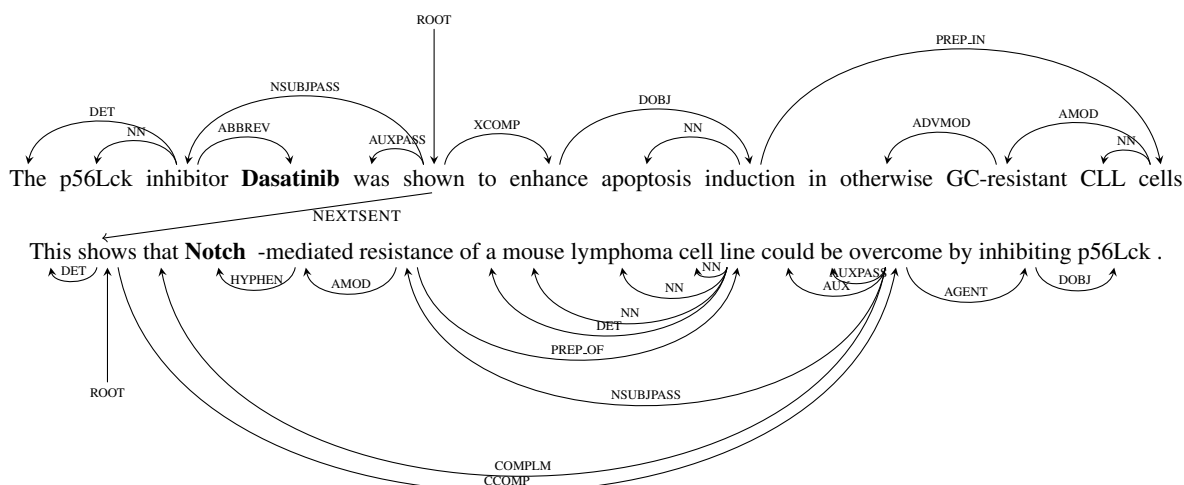


Figure 1: An example document graph for two sentences. Edges represent conventional intra-sentential dependencies, as well as connections between the roots of adjacent sentences (NEXTSENT). For simplicity, we omit edges between adjacent words or representing discourse relations.

3.2 Minimal-Span Candidates

In standard distant supervision, co-occurring entity pairs with known relations are enlisted as candidates of positive training examples. This is reasonable when the entity pairs are within single sentences. In the cross-sentence scenario, however, this would risk introducing too many wrong examples. Consider the following two sentences: *Since amuvatinib inhibits **KIT**, we validated **MET** kinase inhibition as the primary cause of cell death. Additionally, **imatinib** is known to inhibit **KIT**.* The mention of drug-gene pair *imatinib* and *KIT* (in bold) span two sentences, but the same pair also co-occur in the second sentence alone. In general, one might find co-occurring entity pairs in a large text span, where the same pairs also co-occur in a smaller text span that overlaps with the larger one. In such cases, if there is a relation between the pair, mostly likely it is expressed in the smaller text span when the entities are closer to each other.

This motivates us to define that an co-occurring entity pair has the *minimal span* if there does not exist another overlapping co-occurrence of the same pair where the distance between the entity mentions is smaller. Here, the distance is measured in the number of consecutive sentences between the two entities. Experimentally, we compared extraction with or without the restriction to minimal-span candidates, and show that the former led to much higher extraction accuracy.

3.3 Document Graph

To derive features for entity pairs both within and across sentences, DISCREX introduces a *document graph* with nodes representing words and edges representing intra- and inter-sentential relations such as dependency, adjacency, and discourse relations. Figure 1 shows an example document graph spanning two sentences. Each node is labeled with its lexical item, lemma, and part-of-speech. We used a conventional set of intra-sentential edges: typed, collapsed Stanford dependencies derived from syntactic parses (de Marneffe et al., 2006). To mitigate parser errors, we also add edges between adjacent words.

As for inter-sentential edges, a simple but intuitive approach is to add an edge between the dependency roots of adjacent sentences: if we imagined that each sentence participated as a node in a type of discourse dependency tree, this represents a simple right-branching baseline. To gather a finer grained representation of rhetorical structure, we ran a state-of-the-art discourse parser (Wang and Lan, 2015) to identify discourse relations, which returned a set of labeled binary relations between spans of words. We found the shortest path between any word in the first span and any word in the second span using only dependency and adjacent sentence edges, and added an edge labeled with the discourse relation between these two words. Another source of potentially cross-sentence links comes from coreference. We generated coreference relations using the Stanford Coreference systems (both statistical and deter-

ministic) (Lee et al., 2011; Recasens et al., 2013; Clark and Manning, 2015), and added edges from anaphora to their antecedents.

We also considered a special case of cross-sentence relation extraction by augmenting single-sentence candidates with coreference (Koch et al., 2014; Augenstein et al., 2016). Namely, extraction is still conducted within single sentences, yet entity linking is extended to consider all coreference mentions for a relation argument. However, this did not produce significantly more candidates (2% more for positive examples), most of which were not cross-sentence ones (only 1%).

3.4 Features

Dependency paths have been established as a particularly effective source for relation extraction features (Mintz et al., 2009). DISCREX generalizes this idea by defining feature templates over paths in the document graph, which may contain interleaving edges of various types (dependency, word and sentence adjacency, discourse relation). Dependency paths provide interpretable and generalizable features but are subject to parser error. One error mitigation strategy is to add edges between adjacent words, allowing multiple paths between entities.

Feature extraction begins with a pair of entities in the document graph that potentially are connected by a relation. We begin by finding a path between the entities of interest, and extract features from that path.

Over each such path, we explore a number of different features. Below, we assume that each path is a sequence of nodes and edges $(n_1, e_1, n_2, \dots, e_{L-1}, n_L)$, with n_1 and n_L replaced by special entity marker nodes.⁶

Whole path features We extract four binary indicator features for each whole path, with nodes n_i represented by their lexical item, lemma, part-of-speech tag, or nothing. These act as high precision but low recall indicators of useful paths.

Path n-gram features A more robust and generalizable approach is to consider a sliding window along each path. For each position i , we extract n -gram ($n = 1-5$) features starting at each node (n_i , then $n_i \cdot e_i$ and so on until $n_i \cdot e_i \cdot n_{i+1} \cdot e_{i+1} \cdot n_{i+2}$) and each edge (e_i up to $e_i \cdot n_{i+1} \cdot e_{i+1} \cdot n_{i+2} \cdot e_{i+2}$).

⁶ This prevents our method from memorizing the entities in the original knowledge base.

Again, each node could be represented by its lexical item, lemma, or part of speech, leading to 27 feature templates. We add three more feature templates using only edge labels (e_i ; $e_i \cdot e_{i+1}$; and $e_i \cdot e_{i+1} \cdot e_{i+2}$) for a total of 30 feature templates.

3.5 Multiple paths

Most prior work has only looked at the single shortest path between two entities. When authors use consistent lexical and syntactic constructions, and when the parser finds the correct parse, this approach works well. Real data, however, is quite noisy.

One way to mitigate errors and be robust against noise is to consider multiple possible paths. Given a document graph with arcs of multiple types, there are often multiple paths between nodes. For instance, we might navigate from the gene to the drug using only syntactic arcs, or only adjacency arcs, or some combination of the two. Considering such variations gives more opportunities to find commonalities between seemingly disparate language.

We explore varying the number of shortest paths, N , between the nodes in the document graph corresponding to the relevant entities. By default, all edge types have an equal weight of 1, except edges between adjacent words. Empirically, penalizing adjacency edges led to substantial benefits, though including adjacency arcs was important for benefits from multiple paths. This suggests that the parser produces valuable information, but that we should have a back-off strategy for accommodating parser errors.

3.6 Evaluation

There is no gold annotated dataset in distant supervision, so evaluation typically resorts to two strategies. One strategy uses held-out samples from the training dataset, essentially treating the noisy annotation as gold standard. This has the advantage of being automatic, but could produce biased results due to false negatives (i.e., entity pairs not known to have the relation might actually have the relation). Another strategy reports absolute recall (number of extractions from all unlabeled text), as well as estimated precision by manually annotating extraction samples from general text. We conducted both types of evaluation in the experiments.

Disease	Gene	Variant	Description	Effect	Association_1	Therapeutic context_1
ALL	ABL1	T315A	missense mutation	gain-of-function	response	nilotinib, ponatinib
ALL	ABL1	T315I	missense mutation	gain-of-function	response	ponatinib
ALL	ABL1	F317L/V/I/C	missense mutation	gain-of-function	response	nilotinib, ponatinib
ALL	ABL1	F359V/C/I	missense mutation	gain-of-function	response	dasatinib, ponatinib
CML	ABL1	T315A	missense mutation	gain-of-function	response	nilotinib, bosutinib, ponatinib
CML	ABL1	T315I	missense mutation	gain-of-function	response	ponatinib
CML	ABL1	F317L/V/I/C	missense mutation	gain-of-function	response	nilotinib, bosutinib, ponatinib
CML	ABL1	F359V/C/I	missense mutation	gain-of-function	response	dasatinib, bosutinib, ponatinib
ALL	ABL1	Y253H	missense mutation	gain-of-function	response	dasatinib, ponatinib
ALL	ABL1	E255K/V	missense mutation	gain-of-function	response	dasatinib, ponatinib

Figure 2: Sample rows from the Gene Drug Knowledge Database. Our current work focuses on two important columns: gene, and therapeutic context (drug).

4 Experiments

We consider the task of extracting drug-gene interactions from biomedical literature. A drug-gene interaction is broadly construed as an association between the drug efficacy and the gene status. The status includes mutations and activity measurements (e.g., overexpression). For simplicity, we only consider the relation at the drug-gene level, without distinguishing among details such as drug dosage or distinct gene status.

4.1 Knowledge Base

We used the Gene Drug Knowledge Database (GDKD) (Dienstmann et al., 2015) for distant supervision. Figure 2 shows a snapshot of the dataset. Each row specifies a gene, some drugs, the fine-grained relations (e.g., sensitive), the gene status (e.g., mutation), and some supporting article IDs. In this paper, we only consider the coarse drug-gene association and ignore the other fields.

4.2 Unlabeled Text

We obtained biomedical literature from PubMed Central⁷, which as of early 2015 contained about 960,000 full-text articles. We preprocessed the text using SPLAT (Quirk et al., 2012) to conduct tokenization, part-of-speech tagging, and syntactic parsing, and obtained Stanford dependencies (de Marneffe et al., 2006) using Stanford CoreNLP (Manning et al., 2014). We used the entity taggers from Literome (Poon et al., 2014) to identify drug and gene mentions.

4.3 Candidate Selection

To avoid unlikely candidates such as entity pairs far apart in the document, we consider entity pairs within K consecutive sentences. $K = 1$ corresponds to extraction within single sentences. For cross-sentence extraction, we chose $K = 3$ as it

⁷<http://www.ncbi.nlm.nih.gov/pmc/>

Number of Candidates	$K = 1$	$K = 3$
Unique Pairs	169,168	332,969
Instances	1,724,119	3,913,338
Matching GDKD	58,523	87,773

Table 1: Statistics for drug-gene interaction candidates in PubMed Central articles: unique pairs, instances, instances with known relations in Gene Drug Knowledge Database (GDKD).

doubled the number of overall candidates, while being reasonably small so as not to introduce too many unlikely ones. Table 1 shows the statistics of drug-gene interaction candidates identified in PubMed Central articles. For $K = 3$, there are 87,773 instances for which the drug-gene pair has known associations in Gene Drug Knowledge Database (GDKD), which are used as positive training examples. Note that these only include minimal-span candidates (Section 3.2). Without the restriction, there are 225,520 instances matching GDKD, though many are likely false positives.

4.4 Classifier

Our classifiers were binary logistic regression models, trained to optimize log-likelihood with an ℓ_2 regularizer. We used a weight of 1 for the regularizer; the results were not very sensitive to the specific value. Parameters were optimized using L-BFGS (Nocedal and Wright, 2006). Rather than explicitly mapping each feature to its own dimension, we hashed the feature names and retained 22 bits (Weinberger et al., 2009). Approximately 4 million possible features seemed to suffice for our problem: fewer bits produced degradations, but more bits did not lead to improvements.

4.5 Automatic Evaluation

To evaluate the impact of features, we conducted five-fold cross validation, by treating the positive

Features	Single-Sent.	Cross-Sent.
Base	81.3	81.7
3 paths	85.4	85.5
+coref	85.0	84.7
+disc	—	84.6
+coref+disc	—	84.5
10 paths	87.0	86.6
+coref	86.5	85.9
+disc	—	86.5
+coref+disc	—	85.9

Table 2: Average test accuracy in five-fold cross-validation. Cross-sentence extraction was conducted within a sliding window of 3 sentences using minimal-span candidates. *Base* only used the shortest path to construct features. *3 paths* and *10 paths* gathered features from the top three or ten shortest paths, assigning uniform weights to all edges except adjacency, which had a weight of 16. *+coref* adds edges for the relations predicted by Stanford Coreference. *+disc* adds edges for the predicted rhetorical relations by a state-of-the-art discourse parser (Wang and Lan, 2015).

and negative examples from distant supervision as gold annotation. To avoid train-test contamination, all instances from a document are assigned to the same fold. We then evaluated the average test performance across folds. Since our datasets were balanced by design (Section 3.1), we simply reported accuracy. As discussed before, the results could be biased by the noise in annotation, but this automatic evaluation enables an efficient comparison of various design choices.

First, we set out to investigate the impact of edge types and path number. We set the weight for adjacent-word edges to 16, to give higher priority to other edge types (weight 1) that are arguably more semantics-related. Table 2 shows the average test accuracy for single-sentence and cross-sentence extraction with various edge types and path numbers. Compared to extraction within single sentences, cross-sentence extraction attains a similar accuracy, even though the recall for the latter is much higher (Table 1).

Adding more paths other than the shortest one led to a substantial improvement in accuracy. The gain is consistent for both single-sentence and cross-sentence extraction. This is surprising, as prior methods often derive features from the short-

Paths	Adj. Wt.	Single-Sent.	Cross-Sent.
	1	82.2	82.1
3	4	85.0	84.9
	16	85.4	85.5
	64	85.1	85.0
10	1	85.7	83.6
	4	87.2	86.7
	16	87.0	86.6
	64	87.0	86.6
30	1	87.6	85.4
	4	88.0	87.5
	16	87.5	87.2
	64	87.5	87.2

Table 3: Average test accuracy in five-fold cross-validation. Uniform weights are used, except for adjacent-word edges.

est dependency path alone.

Adding discourse relations, on the other hand, consistently led to a small drop in performance, especially when the path number is small. Upon manual inspection, we found that Stanford Coreference made many errors in biomedical text, such as resolving a dummy pronoun with a nearby entity. In hindsight, this is probably not surprising: state-of-the-art coreference systems are optimized for newswire domain and could be ill-suited for scientific literature (Bell et al., 2016). We are less certain about why discourse parsing didn’t seem to help. There are clearly examples where extraction errors could have been avoided given rhetorical relations (e.g., when the sentence containing the second entity starts a new topic). We leave more in-depth investigation to future work.

Next, we further evaluated the impact of path number and adjacency edge weight. Only dependency and adjacency edges were included in these experiments. Table 3 shows the results. Penalizing adjacency produces large gains; a harsh penalty is particularly helpful with fewer paths. These results support the hypothesis that dependency edges are usually more meaningful for relation extraction than word adjacency. Therefore, if adjacency edges get the same weights, they might cause some dependency sub-paths drop out of the top K paths, thus degrading performance. When the path number increases, there is a consistent and substantial increase in accuracy, which demonstrates the advantage of allowing adjacency edges

Relations	Single-Sent.	Cross-Sent.
Candidates	169,168	332,969
$p \geq 0.5$	32,028	64,828
$p \geq 0.9$	17,349	32,775
GDKD	162	

Table 4: Unique drug-gene interactions extracted from PubMed Central articles, compared to the manually curated Gene Drug Knowledge Database (GDKD) used for distant supervision. p signifies the output probability. GDKD contains 341 relations, but only 162 have specific drug references usable as distant supervision.

	Gene	Drug
GDKD	140	80
Single-Sent. ($p \geq 0.9$)	4036	311
Single-Sent. ($p \geq 0.5$)	6189	347
Cross-Sent. ($p \geq 0.9$)	5580	338
Cross-Sent. ($p \geq 0.5$)	9470	373

Table 5: Numbers of unique genes and drugs in the Gene Drug Knowledge Database (GDKD) vs. DISCREX extractions.

to interleave with dependency ones. This presumably helps address syntactic parsing errors, among other things. The importance of adjacency weights decreases with more paths, but it is still significantly better to penalize adjacency edges.

In the experiments mentioned above, cross-sentence extraction was conducted using minimal-span candidates only. We expected that this would provide a reasonable safeguard to filter out many unlikely candidates. As empirical validation, we also conducted experiments on cross-sentence extraction without the minimal-span restriction, using the base model. Test accuracy dropped sharply from 81.7% to 79.1% (not shown in the table).

4.6 PubMed-Scale Extraction

Our ultimate goal is to extract knowledge from all available text. First, we retrained DISCREX on all available distant-supervision data, not restricting to a subset of the folds as in the automatic evaluation. We used the systems performing best on automatic evaluation, with features derived from 30 shortest paths between each entity pair, and minimal-span candidates within three sentences

for cross-sentence extraction. We then applied the learned extractors to all PubMed Central articles. We grouped the extracted instances into unique drug-gene pairs. The classifier output a probability for each instance. The maximum probability of instances in a group was assigned to the relation as a whole. Table 4 shows the statistics of extracted relations by varying the probability threshold. Cross-sentence extraction obtained far more unique relations compared to single-sentence extraction, improving absolute recall by 89-102%. Table 5 compares the number of unique genes and drugs. DISCREX extractions cover far more genes and drugs compared to GDKD, which bode well for applications in precision medicine.

4.7 Manual Evaluation

Automatic evaluation accuracies can be overly optimistic. To assess the true precision of DISCREX, we also conducted manual evaluation on extracted relations. Based on the automatic evaluation, the accuracy is similar for single-sentence and cross-sentence extraction. So we focused on the latter. We randomly sampled extracted relation instances and asked two researchers knowledgeable in precision medicine to evaluate their correctness. For each instance, the annotators were provided with the provenance sentences where the drug-gene pair were highlighted. The annotators assessed in each case whether some relation was mentioned for the given pair.

A total of 450 instances were judged: 150 were sampled randomly from all candidates (random baseline), 150 from the set of instances with probability no less than 0.5, and 150 with probability no less than 0.9. From each set, we randomly selected 50 relations for review by both annotators. The two annotators agreed on 133 of 150. After review, all disagreements were resolved, and each annotator judged an additional set of 50 relation instances, this time without overlap.

Table 6 showed the sample precision and percentage of errors due to entity linking vs. relation extraction. With either classification threshold, cross-sentence extraction clearly outperformed the random baseline by a wide margin. Not surprisingly, the higher threshold of 0.9 led to higher precision. Interestingly, a significant portion of errors stems from mistakes in entity linking, as has been observed in prior work (Poon et al., 2015). Improved entity linking, either alone or joint with re-

	Prec.	Entity Err.	Relation Err.
<i>Single-sentence extractions</i>			
Random	31	52	17
$p \geq 0.5$	61	25	15
$p \geq 0.9$	71	13	15
<i>Cross-sentence extractions</i>			
Random	23	50	27
$p \geq 0.5$	57	20	23
$p \geq 0.9$	61	13	26

Table 6: Sample precision and error percentage: comparison between the single sentence and cross-sentence extraction models at various thresholds. Single sentence extraction is slightly better at all thresholds, at the expense of substantially lower recall: a reduction of 40% or more in terms of unique interactions.

lation extraction, is an important future direction.

Based on these estimates, DISCREX extracted about 37,000 correct unique interactions at the threshold of 0.5, and about 20,000 at the threshold of 0.9. In both cases, it expanded the Gene Drug Knowledge Base by two orders of magnitude.

We also performed manual evaluation in the single-sentence setting. As in the automatic evaluation, single-sentence precisions are similar though slightly higher at all thresholds. This suggests that the candidate set is cleaner and the resulting predictions are more accurate. However, the resulting recall is substantially lower, dropping by 46% at a threshold of 0.5, and by 40% at a threshold of 0.9.

5 Conclusion

We present the first approach for applying distant supervision to cross-sentence relation extraction, by adopting a document-level graph representation that incorporates both intra-sentential dependencies and inter-sentential relations such as adjacency and discourse relations. We conducted both automatic and manual evaluation on extracting drug-gene interactions from biomedical literature. With cross-sentence extraction, our DISCREX system doubled the yield of unique interactions, while maintaining the same accuracy. Using distant supervision, DISCREX improved the coverage of the Gene Drug Knowledge Database (GDKD) by two orders of magnitude, without requiring annotated examples.

Future work includes: further exploration of features; improved integration with coreference and discourse parsing; combining distant supervision with active learning and crowd sourcing; evaluate the impact of extractions to precision medicine; applications to other domains.

References

- Isabelle Augenstein, Diana Maynard, and Fabio Ciravegna. 2016. Distantly supervised web relation extraction for knowledge base population. *Semantic Web*, 7:335–349.
- Orli Bahcall. 2015. Precision medicine. *Nature*, 526:335.
- Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India. AAAI Press.
- Dane Bell, Gustave Hahn-Powell, Marco A. Valenzuela-Escarcega, and Mihai Surdeanu. 2016. An investigation of coreference phenomena in the biomedical domain. In *Proceedings of LREC*, pages 177–183.
- Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 10–18, Boulder, Colorado, June. Association for Computational Linguistics.
- Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore, August. Association for Computational Linguistics.
- Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 837–846, Atlanta, Georgia, June. Association for Computational Linguistics.
- Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1415, Beijing, China, July. Association for Computational Linguistics.

- M. W. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 509–516, Madison, WI. AAAI Press.
- Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. 2000. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118:69–113.
- Dipanjan Das and Noah A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1435–1444, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. ELRA.
- Rodrigo Dienstmann, In Sock Jang, Brian Bot, Stephen Friend, and Justin Guinney. 2015. Database of genomic biomarkers for cancer drugs and clinical targetability in solid tumors. *Cancer Discovery*, 5:118–123.
- Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.
- Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, Maryland, June. Association for Computational Linguistics.
- Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using sub-graph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Lisbon, Portugal, September. Association for Computational Linguistics.
- Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 848–855, Prague, Czech Republic, June. Association for Computational Linguistics.
- Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 289–299, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland, June. Association for Computational Linguistics.
- Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun’ichi Tsujii. 2009. Overview of BioNLP’09 Shared Task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado, June. Association for Computational Linguistics.
- Mitchell Koch, John Gilmer, Stephen Soderland, and S. Daniel Weld. 2014. Type-aware distantly supervised relation extraction with linked arguments. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1891–1901. Association for Computational Linguistics.
- Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford’s multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.
- Daniel Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press, Cambridge, Massachusetts, November.

- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August. Association for Computational Linguistics.
- Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of BioNLP Shared Task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7, Sofia, Bulgaria, August. Association for Computational Linguistics.
- J. Nocedal and S. Wright. 2006. *Numerical Optimization*. Springer, New York, NY.
- Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 756–766, Denver, Colorado, May–June. Association for Computational Linguistics.
- Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI. ACL.
- Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore. ACL.
- Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821, Los Angeles, California, June. Association for Computational Linguistics.
- Hoifung Poon, Chris Quirk, Charlie DeZiel, and David Heckerman. 2014. Literome: Pubmed-scale genomic knowledge base in the cloud. *Bioinformatics*, 30(19):2840–2842.
- Hoifung Poon, Kristina Toutanova, and Chris Quirk. 2015. Distant supervision for cancer pathway extraction from text. In *Pacific Symposium of Biocomputing*, pages 121–131, Big Island of Hawaii.
- Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wentau Yih, Colin Cherry, and Lucy Vanderwende. 2012. MSR SPLAT, a language analysis toolkit. In *Proceedings of the Demonstration Session at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 21–24, Montréal, Canada, June. Association for Computational Linguistics.
- Dragomir R. Radev and Rada Mihalcea. 2008. Networks and natural language processing. *AI Magazine*, 29(3):16–28.
- Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501, Cambridge, MA, October. Association for Computational Linguistics.
- Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977, Singapore, August. Association for Computational Linguistics.
- Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 627–633, Atlanta, Georgia, June. Association for Computational Linguistics.
- Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1–12, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.
- Rebecca Sharp, Peter Jansen, Mihai Surdeanu, and Peter Clark. 2015. Spinning straw into gold: Using free text to train monolingual alignment models for non-factoid question answering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 231–237, Denver, Colorado, May–June. Association for Computational Linguistics.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Sydney, Australia, July. Association for Computational Linguistics.

- Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Cambridge, MA, October. Association for Computational Linguistics.
- Mihai Surdeanu and Heng Ji. 2014. Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population evaluation. In *Proceedings of the TAC-KBP 2014 Workshop*, pages 1–15, Gaithersburg, Maryland, USA.
- Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea, July. Association for Computational Linguistics.
- Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5, Denver, Colorado, June. Association for Computational Linguistics.
- Kumutha Swampillai and Mark Stevenson. 2011. Extracting relations within and across sentences. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 25–32, Hissar, Bulgaria, September. RANLP 2011 Organising Committee.
- Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoi-fung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, September. Association for Computational Linguistics.
- Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoi-fung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1434–1444, Berlin, Germany, August. Association for Computational Linguistics.
- Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 17–24, Beijing, China, July. Association for Computational Linguistics.
- Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pages 1113–1120, New York, NY, USA. ACM.
- Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07*, pages 41–50, New York, NY, USA. ACM.
- Hao Zheng, Zhoujun Li, Senzhang Wang, Zhao Yan, and Jianshe Zhou. 2016. Aggregating inter-sentence information to enhance relation extraction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16*, pages 3108–3114. AAAI Press.

Noise Mitigation for Neural Entity Typing and Relation Extraction

Yadollah Yaghoobzadeh* and Heike Adel* and Hinrich Schütze

* *These authors contributed equally to this work*

Center for Information and Language Processing

LMU Munich, Germany

yadollah|heike@cis.lmu.de

Abstract

In this paper, we address two different types of noise in information extraction models: noise from distant supervision and noise from pipeline input features. Our target tasks are entity typing and relation extraction. For the first noise type, we introduce multi-instance multi-label learning algorithms using neural network models, and apply them to fine-grained entity typing for the first time. Our model outperforms the state-of-the-art supervised approach which uses global embeddings of entities. For the second noise type, we propose ways to improve the integration of noisy entity type predictions into relation extraction. Our experiments show that probabilistic predictions are more robust than discrete predictions and that joint training of the two tasks performs best.

1 Introduction

Knowledge bases (KBs) are important resources for natural language processing tasks like question answering and entity linking. However, KBs are far from complete (e.g., Socher et al. (2013)). Therefore, methods for automatic knowledge base completion (KBC) are beneficial. Two subtasks of KBC are *entity typing (ET)* and *relation extraction (RE)*. We address both tasks in this paper.

As in other information extraction tasks, obtaining labeled training data for ET and RE is challenging. The challenge grows as labels become more fine-grained. Therefore, distant supervision (Mintz et al., 2009) is widely used. It reduces the need for manually created resources. Distant supervision assumes that if an entity has a type (resp. two entities have a relationship) in a KB, then all sentences mentioning that entity (resp. those

two entities) express that type (resp. that relationship). However, that assumption is too strong and gives rise to many *noisy* labels. Different techniques to deal with that problem have been investigated. The main technique is multi-instance (MI) learning (Riedel et al., 2010). It relaxes the distant supervision assumption to the assumption that at least one instance of a bag (collection of all sentences containing the given entity/entity pair) expresses the type/relationship given in the KB. Multi-instance multi-label (MIML) learning is a generalization of MI in which one bag can have several labels (Surdeanu et al., 2012).

Most MI and MIML methods are based on hand crafted features. Recently, Zeng et al. (2015) introduced an end-to-end approach to MI learning based on neural networks. Their MI method takes the most confident instance as the prediction of the bag. Lin et al. (2016) further improved that method by taking other instances into account as well; they proposed MI learning based on selective attention as an alternative way of relaxing the impact of noisy labels on RE. In selective attention, a weighted average of instance representations is calculated first and then used to compute the prediction of a bag.

In this paper, we introduce two multi-label versions of MI. (i) *MIML-MAX* takes the maximum instance for each label. (ii) *MIML-ATT* applies, for each label, selective attention to the instances. We apply MIML-MAX and MIML-ATT to fine-grained ET. In contrast to RE, the ET task we consider contains a larger set of labels, with a variety of different granularities and hierarchical relationships. We show that MIML-ATT deals well with noise in corpus-level ET and improves or matches the results of a supervised model based on global embeddings of entities.

The second type of noise we address in this paper influences the integration of ET into RE. It has

been shown that adding entity types as features improves RE models (cf. Ling and Weld (2012), Liu et al. (2014)). However, noisy training data and difficulties of classification often cause wrong predictions of ET and, as a result, noisy inputs to RE. To address this, we propose a joint model of ET and RE and compare it with methods that integrate ET results in a strict pipeline. The joint model performs best. Among the pipeline models, we show that using probabilities instead of binary decisions better deals with noise (i.e., possible ET errors).

To sum up, our contributions are as follows. (i) We introduce new algorithms for MIML using neural networks. (ii) We apply MIML to fine-grained entity typing for the first time and show that it outperforms the state-of-the-art supervised method based on entity embeddings. (iii) We show that a novel way of integrating noisy entity type predictions into a relation extraction model and joint training of the two tasks lead to large improvements of RE performance.

We release code and data for future research.¹

2 Related Work

Noise mitigation for distant supervision. Distant supervision can be used to train information extraction systems, e.g., in relation extraction (e.g., Mintz et al. (2009), Riedel et al. (2010), Hoffmann et al. (2011), Zeng et al. (2015)) and entity typing (e.g., Ling and Weld (2012), Yogatama et al. (2015), Dong et al. (2015)). To mitigate the noisy label problem, multi-instance (MI) learning has been introduced and applied in relation extraction (Riedel et al., 2010; Ritter et al., 2013). Surdeanu et al. (2012) introduced multi-instance multi-label (MIML) learning to extend MI learning for multi-label relation extraction. Those models are based on manually designed features. Zeng et al. (2015) and Lin et al. (2016) introduced MI learning methods for neural networks. We introduce MIML algorithms for neural networks. In contrast to most MI/MIML methods, which are applied in relation extraction, we apply MIML to the task of fine-grained entity typing. Ritter et al. (2013) applied MI on a Twitter dataset with ten types. Our dataset has a larger number of classes or types (namely 102) and input examples, compared to that Twitter dataset and also to the most widely used datasets for evaluating MI (cf. Riedel et al. (2010)). This makes our setup more challenging because of dif-

ferent dependencies and the multi-label nature of the problem. Also, there seems to be a difference between how entity relations and entity types are expressed in text. Our experiments support that hypothesis.

Knowledge base completion (KBC). Most KBC systems focus on identifying triples $R(e_1, r, e_2)$ missing from a KB (Nickel et al., 2012; Bordes et al., 2013; Weston et al., 2013; Socher et al., 2013; Jiang et al., 2012; Riedel et al., 2013; Wang et al., 2014). Work on entity typing or unary relations for KBC is more recent (Yao et al., 2013; Neelakantan and Chang, 2015; Yaghoobzadeh and Schütze, 2015; Yaghoobzadeh et al., 2017). In this paper, we build a KBC system for unary and binary relations using contextual information of words and entities.

Named entity recognition (NER) and typing. NER systems (e.g., Finkel et al. (2005), Collobert et al. (2011)) used to consider only a small set of entity types. Recent work also addresses fine-grained NER (Yosef et al., 2012; Ling and Weld, 2012; Yogatama et al., 2015; Dong et al., 2015; Del Corro et al., 2015; Ren et al., 2016a; Ren et al., 2016b; Shimaoka et al., 2016). Some of this work (cf. Yogatama et al. (2015), Dong et al. (2015)) treats entity segment boundaries as given and classifies mentions into fine-grained types. We make a similar assumption, but in contrast to NER, we evaluate on the corpus-level entity typing task of Yaghoobzadeh and Schütze (2015); thus, we do not need test sentences annotated with context dependent entity types. This task was also used to evaluate embedding learning methods (Yaghoobzadeh and Schütze, 2016).

Entity types for relation extraction. Several studies have integrated entity type information into relation extraction – either coarse-grained (Hoffmann et al., 2011; Zhou et al., 2005) or fine-grained (Liu et al., 2014; Du et al., 2015; Augenstein et al., 2015; Vlachos and Clark, 2014; Yao et al., 2010; Ling and Weld, 2012) entity types. In contrast to most of this work, but similar to Yao et al. (2010), we do not incorporate binary entity type values, but probabilistic outputs. Thus, we allow the relation extraction system to compensate for errors of entity typing. Additionally, we compare this approach to various other possibilities, to investigate which approach performs best. Yao et al. (2010) found that joint training of entity typing and relation extraction is better than a pipeline

¹cistern.cis.lmu.de

model; we show that this result also holds for neural network models and when the number of entity types is large.

3 MIML Learning for Entity Typing

Entity typing (ET) is the task of finding, for each named entity, a set of types or classes that it belongs to, e.g., “author” and “politician” for “Obama”. Our goal is corpus-level prediction of entity types. We use the entity-type information from a KB and annotated contexts of entities in a corpus to estimate $P(t|e)$, the probability that entity e has type t .

More specifically, consider an entity e and $B = \{c_1, c_2, \dots, c_q\}$, the set of q contexts of e in the corpus. Each c_i is an instance of e and since e can have several labels, it is a multi-instance multi-label (MIML) learning problem. We address MIML using neural networks by representing each context as a vector $\vec{c}_i \in \mathbb{R}^h$, and learn $P(t|e)$ from the set of contexts of entity e . In the following, we first describe our MIML algorithms and then explain how \vec{c}_i is computed.

Notations and definitions. Lowercase letters (e.g., e) refer to variables. Lowercase letters with an upper arrow (e.g., \vec{e}) are vectors. We define BCE, binary cross entropy, as follows where y is a binary variable and \hat{y} is a real valued variable between 0 and 1.

$$\text{BCE}(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y)(1 - \log(\hat{y}))\right)$$

3.1 Algorithms

Distant supervision. The basic way to estimate $P(t|e)$ is based on distant supervision with learning the type probability of each c_i individually, by making the assumption that each c_i expresses all labels of e . Therefore, we define the context-level probability function as:

$$P(t|c_i) = \sigma(\vec{w}_t \vec{c}_i + b_t) \quad (1)$$

where $\vec{w}_t \in \mathbb{R}^h$ is the output weight vector and b_t is the bias scalar for type t . The cost function is defined based on binary cross entropy:

$$L(\theta) = \sum_c \sum_t \text{BCE}(y_t, P(t|c)) \quad (2)$$

where y_t is 1 if entity e has type t otherwise 0. To compute $P(t|e)$ at prediction time, i.e., $P_{\text{pred}}(t|e)$,

the context-level probabilities must be aggregated. Average is the usual way of doing that:

$$P_{\text{pred}}(t|e) = \frac{1}{q} \sum_{i=1}^q P(t|c_i) \quad (3)$$

Multi-instance multi-label. The distant supervision assumption is that *all* contexts of an entity with type t are contexts of t ; e.g., we label all contexts mentioning “Barack Obama” with all of his types. Obviously, the labels are incorrect or *noisy* for some contexts. Multi-instance multi-label (MIML) learning addresses this problem. We apply MIML to fine-grained ET for the first time. Our assumption is: if entity e has type t , then there is at least one context of e in the corpus in which e occurs as type t . So, we apply this assumption during training with the following estimation of the type probability of an entity:

$$P(t|e) = \max_{1 \leq i \leq q} P(t|c_i) \quad (4)$$

which means we take the **maximum** probability of type t over all contexts of entity e as $P(t|e)$. We call this approach **MIML-MAX**.

MIML-MAX picks the most confident context for t , ignoring the probabilities of all the other contexts. Apart from missing information, this can be especially harmful if the entity annotations in the corpus are the result of an entity linking system. In that case, the most confident context might be wrongly linked to the entity. So, it can be beneficial to leverage all contexts into the final prediction, e.g., by **averaging** the type probabilities of all contexts of entity e :

$$P(t|e) = \frac{1}{q} \sum_{i=1}^q P(t|c_i) \quad (5)$$

We call this approach **MIML-AVG**. We also propose a combination of the maximum and average, which uses MIML-MAX (Eq. 4) in training and MIML-AVG (Eq. 5) in prediction. We call this approach **MIML-MAX-AVG**.

MIML-AVG treats every context equally which might be problematic since many contexts are irrelevant for a particular type. A better way is to weight the contexts according to their similarity to the types. Therefore, we propose using selective **attention** over contexts as follows and call this approach **MIML-ATT**. MIML-ATT is the multi-label version of the selective attention method proposed in Lin et al. (2016). To compute the type

Model	Train	Prediction
MIML-MAX	MAX	MAX
MIML-AVG	AVG	AVG
MIML-MAX-AVG	MAX	AVG
MIML-ATT	ATT	ATT

Table 1: Different MIML algorithms for entity typing, and the aggregation function they use to get corpus-level probabilities.

probability for e , we define:

$$P(t|e) = \sigma(\vec{w}_t \vec{a}_t + b_t) \quad (6)$$

where $\vec{w}_t \in \mathbb{R}^h$ is the output weight vector and b_t the bias scalar for type t , and \vec{a}_t is the aggregated representation of all contexts c_i of e for type t , computed as follows:

$$\vec{a}_t = \sum_i \alpha_{i,t} \vec{c}_i \quad (7)$$

where $\alpha_{i,t}$ is the attention score of context c_i for type t and $\vec{a}_t \in \mathbb{R}^h$ can be interpreted as the representation of entity e for type t .

$\alpha_{i,t}$ is defined as:

$$\alpha_{i,t} = \frac{\exp(\vec{c}_i \mathbf{M} \vec{t})}{\sum_{j=1}^q \exp(\vec{c}_j \mathbf{M} \vec{t})} \quad (8)$$

where $\mathbf{M} \in \mathbb{R}^{h \times d_t}$ is a weight matrix that measures the similarity of \vec{c} and \vec{t} . $\vec{t} \in \mathbb{R}^{d_t}$ is the representation of type t .

Table 1 summarizes the differences of our MIML methods with respect to the aggregation function they use to get corpus-level probabilities. For optimization of all MIML methods, we use the binary cross entropy loss function,

$$L(\theta) = \sum_e \sum_t \text{BCE}(y_t, P(t|e)) \quad (9)$$

In contrast to the loss function of distant supervision in Eq. 2, which iterates over all *contexts*, we iterate over all *entities* here.

3.2 Context Representation

To produce a high-quality context representation \vec{c} , we use convolutional neural networks (CNNs).

The first layer of the CNN is a *lookup table* that maps each word in c to an embedding of size d . The output of the lookup layer is a matrix $E \in \mathbb{R}^{d \times s}$ (the embedding layer), where s is the context size (a fixed number of words).

The CNN uses n filters of different window widths w to narrowly convolve E . For each of the n filters $H \in \mathbb{R}^{d \times w}$, the result of applying H to matrix E is a feature map $\vec{m} \in \mathbb{R}^{s-w+1}$:

$$m[i] = g(E_{:,i:i+w-1} \odot H) \quad (10)$$

where g is the *relu* function, \odot is the Frobenius product, $E_{:,i:i+w-1}$ are the columns i to $i+w-1$ of E and $1 \leq w \leq k$ are the window widths we consider. Max pooling then gives us one feature for each filter and the concatenation of those features is the CNN representation of c .

As it is shown in the entity typing part of Figure 1, we apply the CNN to the left and right context of the entity mention and the concatenation $\vec{\phi}(c) \in \mathbb{R}^{2n}$ is fed into a multi-layer perceptron (MLP) to get the final context representation $\vec{c} \in \mathbb{R}^h$:

$$\vec{c} = \tanh(\mathbf{W}_h \vec{\phi}(c)) \quad (11)$$

4 Type-aware Relation Extraction

Relation extraction (RE) is mostly defined as finding relations between pairs of entities, for instance, finding the relation “president-of” between “Obama” and “USA”. Given a set of q contexts for an entity pair z , $B = \{c_1, c_2, \dots, c_q\}$ in the corpus, we learn $P(r|z)$, which is the probability of relation r for z . We assume that each z has one relation $r(z)$. Each c_i is represented by a vector $\vec{c}_i \in \mathbb{R}^h$, which is our type-aware representation of context described in Section 4.1.

To learn $P(r|z)$, we use the multi-instance (MI) learning method of Zeng et al. (2015):

$$P(r|c_i) = \text{softmax}(\mathbf{W}_{\text{out}} \vec{c}_i), \quad (12)$$

$$P(r|z) = \max_{1 \leq i \leq q} P(r|c_i)$$

where $P(r|c_i)$ is the probability of relation r for context c_i . The cost function we optimize is:

$$L(\theta) = - \sum_z \log P(r(z)|z)$$

4.1 Context Representation

Similar to our entity typing system, we apply CNNs to compute the context representation $\vec{\phi}(c)$. In particular, we use Adel et al. (2016)’s CNN. It uses an input representation designed for RE. Each sentence is split into three parts: left of the relation arguments, between the relation arguments

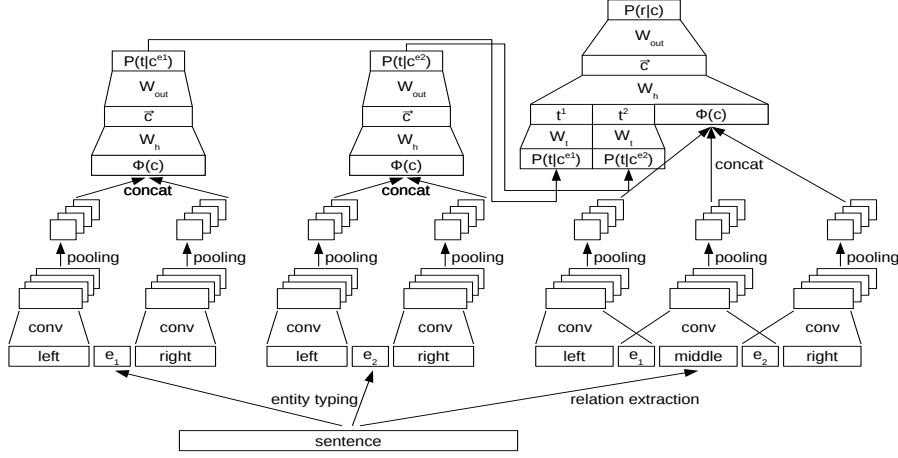


Figure 1: Our architecture for joint entity typing and relation extraction

and right of the relation arguments. The parts “overlap”, i.e., the left (resp. right) argument is included in both left (resp. right) and middle parts. For each of the three parts, convolution and 3-max pooling (Kalchbrenner et al., 2014) is performed. The context representation $\vec{\phi}(c) \in \mathbb{R}^{3 \cdot 3 \cdot n}$ is the concatenation of the pooling results.

4.1.1 Integration of Entity Types

We concatenate the entity type representations $t^1 \in \mathbb{R}^\tau$ and $t^2 \in \mathbb{R}^\tau$ of the relation arguments to the CNN representation of the context, $\vec{\phi}(c)$:

$$\vec{\phi}(c)' = [\vec{\phi}(c) : t^1 : t^2] \quad (13)$$

Our context representation \vec{c} is then:

$$\vec{c} = \tanh(\mathbf{W}_h \vec{\phi}(c)') \quad (14)$$

where $\mathbf{W}_h \in \mathbb{R}^{h \times (3 \cdot 3 \cdot n + 2\tau)}$ is the weight matrix. This is also depicted in Figure 1, right column, third layer from the top: t^1 , t^2 , $\vec{\Phi}(c)$. We calculate t^1 and t^2 from the predictions of the entity typing model with the following transformation:

$$t^k = f(\mathbf{W}_t [P(t_1|c^{e_k}) \dots P(t_T|c^{e_k})]) \quad (15)$$

where c^{e_k} is the context of e_k , $\mathbf{W}_t \in \mathbb{R}^{\tau \times T}$ is a weight matrix (learned from corpus or during training) and f is a function (identity or tanh). With the transformation \mathbf{W}_t , the model can combine predictions for different types to learn better internal representations t^1 and t^2 . The choices of \mathbf{W}_t and f depend on the different representations we investigate and describe in the following.

(1) Pipeline: We integrate entity types into the RE model, using the output of ET in a pipeline

model (see Eq. 15). We test the following representations of t^k , $k \in \{1, 2\}$. **PREDICTED-HIDDEN:** \mathbf{W}_t from Eq. 15 is learned during training and f is tanh. That means that a hidden layer learns representations based on the predictions $P(t_1|c^{e_k}) \dots P(t_T|c^{e_k})$. **BINARY-HIDDEN:** This is the binarization of the input of PREDICTED-HIDDEN, i.e., each probability estimate is converted to 0 or 1 (with a threshold of 0.5). **BINARY:** t^k is the binary vector itself (used by Ling and Weld (2012)). **WEIGHTED:** The columns of matrix \mathbf{W}_t from Eq. 15 are the distributional embeddings of types trained on the corpus (see Section 5.1). f is the identity function.

(2) Joint model: As an alternative to the pipeline model, we investigate integrating entity typing into RE by jointly training both models. We use the architecture depicted in Figure 1. The key difference to the pipeline model PREDICTED-HIDDEN is that we learn $P(t|c)$ and $P(r|c)$ jointly, called **JOINT-TRAIN**. We compare JOINT-TRAIN to other models, including the pipeline models.

During training of JOINT-TRAIN, we compute the cost of the ET model for typing the first entity $L_1(\theta_T)$, the cost for typing the second entity $L_2(\theta_T)$ and the cost of the RE model for assigning a relation to the two entities $L(\theta_R)$. Then, we combine those costs with a weight γ which is tuned on the development set:

$$L(\theta) = \sum_z (L_1(\theta_T) + L_2(\theta_T) + \gamma \cdot L(\theta_R)),$$

$$L_i(\theta_T) = \sum_t \text{BCE}(y_t^{e_i}, P(t|c^{e_i})),$$

$$L(\theta_R) = -\log P(r(z)|z)$$

GOV.GOV_agency.jurisdiction	PPL.PER.children
GOV.us_president.vice_president	PPL.PER.nationality
PPL.deceased.PER.place_of_death	PPL.PER.religion
ORG.ORG.place_founded	PPL.PER.place_of_birth
ORG.ORG_founder.ORGs_founded	NA (no relation)
LOC.LOC.containedby	

Table 2: Selected relations for relation extraction; PPL = people, GOV = government

$P(r|z)$ is computed based on Eq. 12.

Note that based on this equation, the ET parameters are optimized on the contexts of the RE examples, which are a subset of all training examples of ET. However in the pipeline models, ET is trained on the whole training set used for typing. Also note that in JOINT-TRAIN we do not use MIML for the ET part but a distant supervised cost function.

5 Experimental Data, Setup and Results

For entity typing, we use CF-FIGMENT (URL, 2016b), a dataset published by Yaghoobzadeh and Schütze (2015). CF-FIGMENT is derived from a version of ClueWeb (URL, 2016c) in which Freebase entities are annotated using FACC1 (URL, 2016d; Gabrilovich et al., 2013). CF-FIGMENT contains 200,000 Freebase entities that were mapped to 102 FIGER types (Ling and Weld, 2012), divided into train (50%), dev (20%) and test (30%); and a set of 4,300,000 sentences (contexts) containing those entities.

For relation extraction, we first select the ten most frequent relations (plus NA for no relation according to Freebase) of entity pairs in CF-FIGMENT. We ensure that the entity pairs have at least one context in CF-FIGMENT. This results in 5815, 3054 and 6889 unique entity pairs for train, dev and test.² Dev and test set sizes are 124,462 and 556,847 instances. For the train set, we take a subsample of 135,171 sentences. The entity and sentence sets of CF-FIGMENT were constructed to ensure that entities in the entity test set do not occur in the sentence train and dev sets; that is, a sentence was assigned to the train set only if all entities it contains are train entities.¹

5.1 Word, Entity and Type Embeddings

We use 100-dimensional word embeddings to initialize the input layer of ET and RE. Embeddings

²We only assign those entity pairs to test (resp. dev, resp. train) for which both constituting entities are in the ET test (resp. dev, resp. train) set.

are kept fixed during training. Since we need embeddings for words, entities and types in the same space, we process ClueWeb+FACC1 (corpus with entity information) as follows. For each sentence s , we add two copies: s itself, and a copy in which each entity is replaced with its notable type, the most important type according to Freebase. We process train, dev and test this way, but do not replace test entities with their notable type because the types of test entities are unknown in our application scenario. We run word2vec (Mikolov et al., 2013) on the resulting corpus to learn embeddings for words, entities and types. Note that our application scenario is that we are given an unannotated input corpus and our system then extracts entity types and relations from this input corpus to enhance the KB.

5.2 Entity Typing Experiments

Entity context setup. We use a window size of 5 on each side of the entity mentions. Following Yaghoobzadeh and Schütze (2015), we replace other entities occurring in the context with their Freebase notable type mapped to FIGER.

Models. Yaghoobzadeh and Schütze (2015) applied a multi-layer perceptron (MLP) architecture to create context representations. Therefore, we use an MLP baseline to compute the context representation $\vec{\phi}(c)$. The input to the MLP model is a concatenation of context word embeddings. As an alternative to MLP, we also train a CNN (see Section 3.2) to compute context representations. We run experiments with MLP and CNN, each trained with standard distant supervision and with MIML.

EntEmb and FIGMENT baselines. Following Yaghoobzadeh and Schütze (2015), we also learn entity embeddings and classify those embeddings to types, i.e., instead of distant supervision, we classify entities based on aggregated information represented in entity embeddings. An MLP with one hidden layer is used as classifier. We call that model EntEmb. We join the results of EntEmb with our best model (line 13 in Table 3), similar to the joint model (FIGMENT) in Yaghoobzadeh and Schütze (2015).

We use the same **evaluation measures** as Ling and Weld (2012), Yaghoobzadeh and Schütze (2015) and Neelakantan and Chang (2015) for entity typing: precision at 1 ($P@1$), which is the accuracy of picking the most confident type for each entity, micro average F_1 of all entity-type

	$P@1$	F_1	F_1	F_1	MAP
	all	all	head	tail	
1 MLP	74.3	69.1	74.8	52.5	42.1
2 MLP+MIML-MAX	74.7	59.2	50.7	46.8	41.3
3 MLP+MIML-AVG	77.2	70.6	74.9	56.2	45.0
4 MLP+MIML-MAX-AVG	75.2	71.2	76.4	56.0	47.1
5 MLP+MIML-ATT	81.0	72.0	76.9	59.1	48.8
6 CNN	78.4	72.2	77.3	56.3	47.6
7 CNN+MIML-MAX	78.6	62.2	53.5	49.7	46.6
8 CNN+MIML-AVG	80.8	73.5	77.7	59.2	50.4
9 CNN+MIML-MAX-AVG	79.9	74.3	79.2	59.8	53.3
10 CNN+MIML-ATT	83.4	75.1	79.4	62.2	55.2
11 EntEmb	80.8	73.3	79.9	57.4	56.6
12 FIGMENT	81.6	74.3	80.3	60.1	57.0
13 CNN+MIML-ATT+EntEmb	85.4	78.2	83.3	66.2	64.8

Table 3: $P@1$, Micro F_1 for all, head and tail entities and MAP results for entity typing.

assignments and mean average precision (MAP) over types. We could make assignment decisions based on the standard criterion $p > \theta$, $\theta = 0.5$, but we found that tuning θ improves results. For each probabilistic classifier and each type, we set θ to the value that maximizes performance on dev.

Results. Table 3 shows results for $P@1$, micro F_1 and MAP. For F_1 , we report separate results for all, head (frequency higher than 100) and tail (frequency less than 5) entities.

Discussion. The improvement of CNN (6) compared to MLP (1) is not surprising considering the effectiveness of CNNs in finding position independent local features, compared to the flat representation of MLP. Lines 2-5 and 7-10 show the results of different MIML algorithms for MLP and CNN, respectively. Considering micro F1 for all entities as the most importance measure, the trend is similar in both MLP and CNN for MIML methods: ATT > MAX-AVG > AVG > MAX.

MAX is worse than even basic distant supervised models, especially for micro F1. MAX predictions are based on only one context of each entity (for each type), and the results suggest that this is harmful for entity typing. This is in contradiction with the previous results in RE (cf. Zeng et al. (2015)) and suggests that there might be a significant difference between expressing types of entities and relations between them in text. Related to this, MAX-AVG which averages the type probabilities at prediction time improves MAX by a large margin. Averaging the context probabilities seems to be a way to smooth the entity type probabilities. MAX-AVG models are also better than the corresponding models with AVG that train and predict with averaging. This is due to the fact that AVG gives equal weights to all context probabilities both in training and prediction. ATT uses

	ATT			MAX		
	person	author	doctor	person	author	doctor
... /m/024g5w , and DOCTOR into disease will be ...						
... whooping cough , and kidney disease (/m/024g5w 's disease ...						
In 7 , DOCTOR and /m/024g5w write Elements of the ...						
book but his catarrhal bronchitis turned to /m/024g5w 's disease and ...						
It has cured /m/024g5w 's disease that could be traced to ...						
two clinical wards so /m/024g5w can carry on intensive study ...						
/m/024g5w , who once explored LOCATION-COUNTRY and wrote up his ...						
... is /m/024g5w , who is collecting and painstakingly recording ...						

Figure 2: MIML-ATT and MIML-MAX scores for the example entity /m/024g5w.

weighted contexts in both training and prediction and that is probably the reason for its effectiveness over all other MIML algorithms. Overall, using attention (ATT) significantly improves the results of both MLP and CNN models.

CNN+MIML-ATT (10) performs comparable to EntEmb (11), with better micro F1 on all and tail entities and worse MAP and micro F1 on head entities. These two models have different properties, e.g., MIML is also able to type each mention of entities while EntEmb works only for corpus-level typing of entities. (See Yaghoobzadeh and Schütze (2015) for more differences) It is important to note that MIML can also be applied to any entity typing architecture or model that is trained by distant supervision. Due to the lack of large annotated corpora, distant supervision is currently the only viable approach to fine-grained entity typing; thus, our demonstration of the effectiveness of MIML is an important finding for entity typing.

Joining the results of CNN+MIML-ATT with EntEmb (line 13) gives large improvements over each of the single models. It is also consistently better (by more than 3% in all measures) than our baseline FIGMENT (12), which is basically MLP+EntEmb. This improvement is achieved by using CNN instead of MLP for context representation and integrating MIML-ATT. EntEmb is improved by Yaghoobzadeh et al. (2017) by using entity names. We leave the integration of that model to future work.

Example. To show the behavior of MIML-MAX and MIML-ATT, we extract the scores that each method assigns to the labels for each context. A comparison for the example entity “Richard Bright” (MID: /m/024g5w) who is a PERSON, DOCTOR and AUTHOR is shown in Figure 2. Note

that the weights from MIML-ATT (Eq. 8) sum to 1 for each label because of the applied softmax function while the scores from MIML-MAX (Eq. 1) do not. For both methods, the scores for the type PERSON are more equally distributed than for the other types which makes sense since the entity has the PERSON characteristics in every sentence. For the other types, both models seem to be influenced by other entities occurring in the context (e.g., an occurrence with another DOCTOR could indicate that the entity is also a DOCTOR) but also by trigger words such as “write” or “book” for the type AUTHOR or “disease” for the type DOCTOR.

5.3 Relation Extraction Experiments

Models. In our experiments, we compare two state-of-the-art RE architectures: piecewise CNN (Zeng et al., 2015) and contextwise CNN (Adel et al., 2016). We use the publicly available implementation for the piecewise CNN (URL, 2016a) and our own implementation for the contextwise CNN. Both CNNs represent the input words with embeddings and split the contexts based on the positions of the relation arguments. The contextwise CNN splits the input before convolution, the piecewise CNN after convolution. Also, while the piecewise CNN applies a softmax layer directly after pooling, the contextwise CNN feeds the pooling results into a fully-connected hidden layer first. For both models, we use MI learning to mitigate the noise from distant supervision.

Results. The precision recall (PR) curves in Figure 3 show that the contextwise CNN outperforms the piecewise CNN on our RE dataset. We also compare them to a baseline model that does not learn context features but uses only the embeddings of the relation arguments as an input and feeds them into an MLP (similar to the EntEmb baseline for ET). The results confirm that the context features which the CNNs extract are very important, not only for ET but also for RE. Note that the PR curves are calculated on the corpus level and not on the sentence-level, i.e., after aggregating the predictions for each entity pair. Following Ritter et al. (2013), we compute the area A under the PR curves which supports this trend (EntEmb: $A = 0.34$, piecewise CNN: $A = 0.48$, contextwise CNN: $A = 0.50$).

Pipeline vs. joint training. Since the contextwise CNN outperforms the piecewise CNN, we use the contextwise CNN for integrating en-

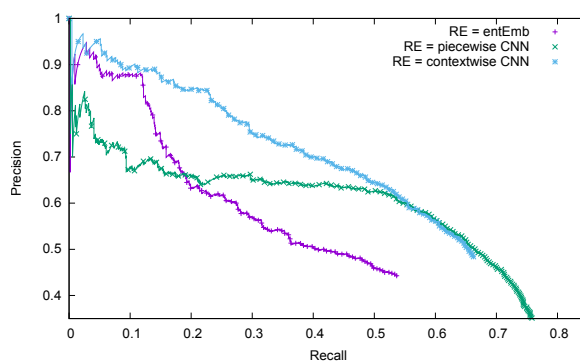


Figure 3: PR curves: relation extraction models

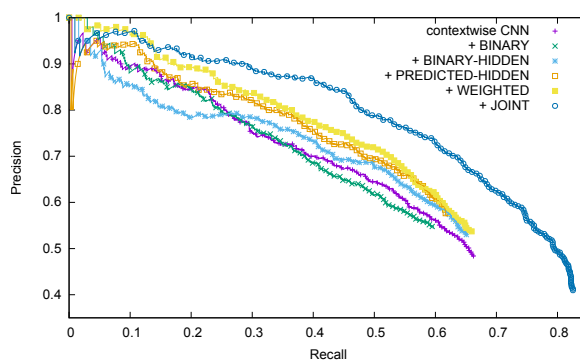


Figure 4: PR curves: type-aware relation extraction models

tity types. Figure 4 shows that the performance on the RE dataset increases when we integrate entity type information into the CNN. The main trend of the PR curves and the areas under them shows the following order of model performances: JOINT-TRAIN > WEIGHTED > PREDICTED-HIDDEN > BINARY-HIDDEN > BINARY.

Discussion. The better performance of our approaches of integrating type predictions into the contextwise CNN (PREDICTED-HIDDEN, WEIGHTED) compared to baseline type integrations (BINARY, BINARY-HIDDEN) shows that probabilistic predictions of an entity typing system can be a valuable resource for RE. With binary types, it is not possible to tell whether one of the selected types had a higher probability than another or whether a type whose binary value is 0 just barely missed the threshold. Probabilistic representations preserve this information. Thus, using probabilistic representations, the RE system can compensate for noise in ET predictions.

WEIGHTED with access to the distributional type embeddings learned from the corpus works better than all other pipeline models. This shows

that our type embeddings can be valuable for RE. JOINT-TRAIN performs better than all pipeline models, even though the ET part in the pipelines is trained on more data. The area of JOINT-TRAIN under the PR curve is $A = 0.66$. A plausible reason is the mutual dependencies of those two tasks which a joint model can better learn than a pipeline model. We can also relate it to better noise mitigation of jointed ET, compared to isolated models.³

Analysis of joint training. In this paragraph, we investigate the joint training in more detail. In particular, we evaluate different variants of it by combining relation extraction with other entity typing approaches: EntEmb and FIGMENT. For joint training with ET-EntEmb, we do not use the context for predicting the types of the relation arguments but only their embeddings. Then, we feed those embeddings into an MLP which computes a representation that we use for the type prediction. This corresponds to the EntEmb model presented in Table 3 (line 11). For joint training with ET-FIGMENT, we compute two different cost functions for entity typing: one for typing based on entity embeddings (see ET-EntEmb above) and one for typing based on an MLP context model. This does not correspond exactly to the FIGMENT model from Table 3 (line 12) which combines an entity embedding and MLP context model as a postprocessing step but comes close. In addition to those two baseline ET models, we also train a version in which both entity typing and relation extraction use EntEmb as their only input features. Figure 5 shows the PR curves for those models. The curve for the model that uses only entity embedding features for both entity typing and relation extraction is much worse than the other curves. This emphasizes the importance of our context model for RE (see also Figure 3), also in combination with joint training. Similarly, the curve for the model with EntEmb as entity typing component has more precision variations than the curves for the other models which use context features for ET. Thus, joint training does not help per se but it is important which models are trained together. The areas under the PR curves show the following model trends: joint with ET-FIGMENT \approx joint as in Figure 1 $>$ joint with ET-EntEmb $>$ joint with ET-EntEmb and RE-EntEmb.

Most improved relations. To identify which

³On the joint dataset, joint training improves MAP for entity typing by about 20% compared to the best isolated model.

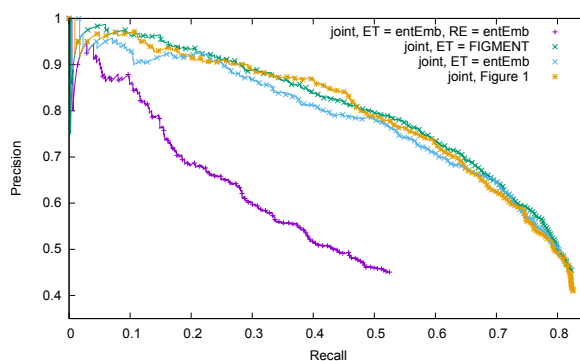


Figure 5: Variants of joint training

relations are improved the most when entity types are integrated, we compare the relation specific F_1 scores of CNN, CNN+WEIGHTED and CNN+JOINT-TRAIN. With WEIGHTED, the relations PPL.deceased_PER.place_of_death and LOC.LOC.containedby are improved the most (from 36.13 to 53.73 and 49.04 to 64.19 F_1 , resp.). JOINT-TRAIN has the most positive impact on PPL.deceased_PER.place_of_death, ORG.ORG.place_founded and GOV.GOV_agency.jurisdiction (from 36.13 to 67.10, 42.38 to 58.51 and 62.26 to 70.41 resp.).

6 Conclusion

In this paper, we addressed different types of noise in two information extraction tasks: entity typing and relation extraction. We presented the first multi-instance multi-label methods for entity typing and showed that it helped to alleviate the noise from distant supervised labels. This is an important contribution because most of the current fine-grained entity typing systems are trained by distant supervision. Our best model sets a new state of the art in corpus-level entity typing. For relation extraction, we mitigated noise from using predicted entity types as features. We compared different pipeline approaches with each other and with our proposed joint type-relation extraction model. We observed that using type probabilities is more robust than binary predictions of types, and joint training gives the best results.

Acknowledgments

This work was supported by DFG (SCHU2246/8-2) and by a Google European Doctoral Fellowship granted to Heike Adel.

References

- Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 828–838, San Diego, California, June. Association for Computational Linguistics.
- Isabelle Augenstein, Andreas Vlachos, and Diana Maynard. 2015. Extracting relations between non-standard entities using distant supervision and imitation learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 747–757, Lisbon, Portugal, September. Association for Computational Linguistics.
- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Irreflexive and hierarchical relations as translations. In *ICML 2013 Workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 868–878, Lisbon, Portugal, September. Association for Computational Linguistics.
- Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1243–1249.
- Lan Du, Anish Kumar, Mark Johnson, and Massimiliano Ciaramita. 2015. Using entity information from a knowledge base to improve relation extraction. In *Australasian Language Technology Association Workshop 2015*, pages 31–38.
- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 363–370, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Evgeniy Gabrilovich, Michael Ringgaard, and Amarag Subramanya. 2013. Facc1: Freebase annotation of cluweb corpora.
- Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA, June. Association for Computational Linguistics.
- Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. 2012. Link prediction in multi-relational graphs using additive models. In *Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data, Boston, USA, November 11, 2012*, pages 1–12.
- Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.
- Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany, August. Association for Computational Linguistics.
- Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, Toronto, Ontario, Canada., July.
- Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2107–2116, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at 1st International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA, May.
- Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August. Association for Computational Linguistics.
- Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge

- base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525, Denver, Colorado, May–June. Association for Computational Linguistics.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *World Wide Web Conference*, pages 271–280.
- Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378, Austin, Texas, November. Association for Computational Linguistics.
- Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1825–1834.
- Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.
- Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *TACL*, 1:367–378.
- Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 69–74, San Diego, CA, June. Association for Computational Linguistics.
- Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 926–934, Lake Tahoe, Nevada, United States., December.
- Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea, July. Association for Computational Linguistics.
- URL. 2016a. Ds_pcnnns (piecewise cnn) code (kang liu). http://www.nlpr.ia.ac.cn/cip/~liukang/liukangPageFile/code/ds_pcnnns-master.zip.
- URL. 2016b. Fimgent data set. <http://cistern.cis.lmu.de/fimgent>.
- URL. 2016c. Lemur project: Clueweb. <http://lemurproject.org/clueweb12>.
- URL. 2016d. Lemur project: Faccl. <http://lemurproject.org/clueweb12/FACCL>.
- Andreas Vlachos and Stephen Clark. 2014. Application-driven relation extraction with limited distant supervision. In *Proceedings of the AHA! Workshop on Information Discovery in Text, Dublin, Ireland, August 23 2014*, pages 1–6.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar, October. Association for Computational Linguistics.
- Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, Lisbon, Portugal, September. Association for Computational Linguistics.
- Yadollah Yaghoobzadeh and Hinrich Schütze. 2016. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246, Berlin, Germany, August. Association for Computational Linguistics.
- Yadollah Yaghoobzadeh, , and Hinrich Schütze. 2017. Multi-level representations for fine-grained typing of knowledge base entities. In *EACL*, Valencia, Spain.

- Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, Cambridge, MA, October. Association for Computational Linguistics.
- Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, AKBC '13*, pages 79–84, San Francisco, California, USA, October.
- Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 291–296, Beijing, China, July. Association for Computational Linguistics.
- Mohamed Amir Yosef, Sandro Bauer, Johannes Hofmann, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters*, pages 1361–1370, Mumbai, India, December. The COLING 2012 Organizing Committee.
- Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, Lisbon, Portugal, September. Association for Computational Linguistics.
- GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Analyzing Semantic Changes in Japanese Loanwords

Hiroya Takamura

Tokyo Institute of Technology
takamura@pi.titech.ac.jp

Ryo Nagata

Konan University
nagata-acl@hyogo-u.ac.jp

Yoshifumi Kawasaki

Sophia University
kyossii@gmail.com

Abstract

We analyze semantic changes in loanwords from English that are used in Japanese (Japanese loanwords). Specifically, we create word embeddings of English and Japanese and map the Japanese embeddings into the English space so that we can calculate the similarity of each Japanese word and each English word. We then attempt to find loanwords that are semantically different from their original, see if known meaning changes are correctly captured, and show the possibility of using our methodology in language education.

1 Introduction

We often come across advertisements that have extravagant images. In Japan, such images are usually accompanied by the following sentence¹:

この 画像は イメージ です
kono gazō-wa imēji desu
this image-TOP image COP
“This image is an image.”

This sentence sounds like a nonsense tautology, but actually means *this image is only for illustrative purposes and may differ from the actual product*. Both *gazō* and *imēji* are Japanese words, each meaning *image*. However, the latter is a loanword from English, i.e., *image*². In the sentence above, *imēji*, the loanword for image, is closer in meaning to the word *impression*, and it makes the sentence roughly mean *this image is just an impres-*

¹TOP and COP respectively mean a topic marker and a copula in interlinear glossed text (IGT) representation. The last line is a literal translation of the Japanese sentence.

²Note that although *gazō* is also from ancient Chinese, we focus on loanwords from English, which are usually written in *katakana* letters in Japanese.

sion that you might have on this product. What happens in this seeming tautology is that the loanword changes meaning; i.e., the sense of the loanword deviates from the sense of its original word.

Loanwords from English occupy an important place in the Japanese language. It is reported that approximately 8% of the vocabulary of contemporary Japanese consists of loanwords from English (Barrs, 2013). One noteworthy characteristics of loanwords in Japanese is that their meanings are often different from their original words, as in the above example. Indeed, the meanings of loanwords in any language are not generally the same as those in the language, but according to Kay (1995), Japanese has particularly a strong tendency of changing the meanings of loanwords; Kay argued that in Japan *there is no deep cultural motivation to protect their original meaning*. Daulton (2009) also argued that Japanese loanwords are malleable in terms of meanings. Thus, Japanese loanwords would be an interesting subject to work on in the study of meaning change.

Japanese loanwords from English are also important in language education (Barrs, 2013). Japanese learners of English often make mistakes in using English words that have corresponding loanwords in Japanese but with very different meanings. By contrast, learners are able to make better use of a loanword in conversation if they know that its meaning is the same as that of the original. It is thus important to know which loanwords are semantically different from their original and which are not.

With this background in mind, we work on Japanese loanwords derived from English. Since the word embedding vectors (or simply, embeddings), which have become very popular recently, are powerful tools for dealing with word meanings, we use them to analyze Japanese loanwords. Specifically, we create word embeddings of En-

glish and Japanese, and map the Japanese embeddings into the English space so that we can calculate the similarity of each Japanese word and each English word. We then attempt to find loanwords that are semantically different from their original, see if known meaning changes are correctly captured, and show the possibility of using our methodology in language education.

In this paper, we use the term *semantic change* or *meaning change* in a broad sense. Some loanwords are semantically different from the original words because the loanwords or the original words semantically changed after they were introduced into Japanese or because only one of the multiple senses of the original words were introduced. Moreover, some loanwords did not come directly from English, but from words in other languages, which later became English words. Thus, in this paper, the terms semantic change or meaning change cover all of these semantic differences.

2 Related Work

Japanese loanwords have attracted much interest from researchers. Many interesting aspects of Japanese loanwords are summarized in a book written by Irwin (2011). In the field of natural language processing, there have been a number of efforts to capture the behavior of Japanese loanwords including the phonology (Blair and Ingram, 1998; Mao and Hulden, 2016) and segmentation of multi-word loanwords (Breen et al., 2012). The rest of this section explains the computational approaches to semantic changes or variations of words. In particular, there are mainly two different phenomena, namely diachronic change and geographical variation.

Jatowt and Duh (2014) used conventional distributional representations of words, i.e., bag-of-context-words, calculated from Google Book (Michel et al., 2011)³ to analyze the diachronic meaning changes of words. They also attempted to capture the change in sentiment of words across time. Kulkarni et al. (2014) used distributed representations of words (or word embeddings), instead of the bag-of-context-words used by Jatowt and Duh, to capture meaning changes of words and in addition used the change point detection technique to find the point on the timeline where the meaning change occurred. Hamil-

³<https://books.google.com/ngrams/datasets>

ton et al. (2016) also used distributed representations for the same purpose and attempted to reveal the statistical laws of meaning change. They compared the following three methods for creating word embedding: positive pointwise mutual information (PPMI), low-dimensional approximation of PPMI obtained through singular value decomposition, and skip-gram with negative sampling. They suggested that the skip-gram with negative sampling is a reasonable choice for studying meaning changes of words. We decided to follow their work and use the skip-gram with negative sampling to create word embeddings.

Bamman et al. (2014) used a similar technique to study differences in word meanings ascribed to geographical factors. They succeeded in correctly recognizing some dialects of English within the United States. Kulkarni et al. (2016) also worked on geographic variations in languages.

With some modification, the methods used in the literature (Kulkarni et al., 2014; Hamilton et al., 2016) can be applied to loanword analysis.

3 Methodology

We use word embeddings to analyze the semantic changes in Japanese loanwords from the corresponding English. Among the methods of analysis, we chose to use the skip-gram with negative sampling for the reason discussed in Section 2 with reference to Hamilton et al.’s work (2016).

First, we create word embeddings for two languages. We then calculate the similarity or dissimilarity between the embedding (or vector) of a word in a language (say, Japanese) and the embedding of a word in another language (say, English). For this purpose, words in the two languages need to be represented in the same vector space with the same coordinates. There are a number of methods for this purpose (Gouws et al., 2015; Zou et al., 2013; Faruqui and Dyer, 2014; Mikolov et al., 2013a). Among them, we choose the simplest and most computationally efficient one proposed by Mikolov et al. (2013a), where it is assumed that embeddings in one language can be mapped into the vector space of another language by means of a linear transformation represented by W . Suppose we are given trained word embeddings of the two languages and a set of seed pairs of embedding vectors $\{(x_i, z_i) | 1 \leq i \leq n\}$, each of which is a pair of a vector in one language and a vector in the other language that are translation equiva-

lents of each other. The transformation matrix W is obtained by solving the following minimization problem :

$$\min_W \sum_{i=1}^n \|Wx_i - z_i\|^2, \quad (1)$$

where, in our case, x_i is the embedding of a Japanese seed word and z_i is the embedding of its English counterpart. Thus, the Japanese word embeddings are mapped into the English vector space so that the embeddings of the words in each seed pair should be as close to each other as possible. Although Hamilton et al. (2016) preserved cosine similarities between embedding vectors by adding the orthogonality constraint (i.e., $W^T W = I$, where I is the identity matrix) when they aligned English word embeddings of different time periods, we do not adopt this constraint for two reasons. The first reason is that since we need an inter-language mapping instead of across-time mappings of the same language, the orthogonality constraint would degrade the quality of the mapping; the two spaces might be so different that even the best rotation represented by an orthogonal matrix would leave much error between corresponding words. The second reason is that we do not need to preserve cosine similarities between words in mapping embedding vectors, because we do not use the cosine similarities between mapped embedding vectors of Japanese words.

After mapping the Japanese word embeddings to the English vector space, we calculate the cosine similarity between each Japanese loanword and its original English word. If the cosine similarity is low for a pair of words, the meaning of the Japanese loanword is different from that of its original English word.

4 Empirical Evaluations

Since it is generally difficult to evaluate methods for capturing semantic changes in words, we conduct a number of quantitative and qualitative evaluations from different viewpoints.

4.1 Data and Experimental Settings

The word embeddings of English and Japanese were obtained via the skip-gram with negative sampling (Mikolov et al., 2013b)⁴ with different dimensions as shown in the result. The data

⁴<https://code.google.com/archive/p/word2vec/> with options “-window 5 -sample 1e-4 -negative 5 -hs 0 -cbow 0 -iter 3”

used for this calculation was taken from Wikipedia dumps⁵ as of June 2016 for each language; the text was extracted by using wp2txt (Hasebe, 2006)⁶, non-alphabetical symbols were removed, and noisy lines such as the ones corresponding to the infobox were filtered out⁷. We performed word segmentation on the Japanese Wikipedia data by using the Japanese morphological analyzer MeCab (Kudo et al., 2004)⁸ with the neologism dictionary, NEologd⁹, so that named entities would be recognized correctly.

The list of Japanese loanwords was obtained from Wiktionary¹⁰. Only one-word entries were used and some errors were corrected, resulting in 1,347 loanwords from English¹¹.

We extracted seed word pairs from an English-Japanese dictionary, edict (Breen, 2000)¹²; these were used in the minimization problem expressed by Equation (1). Specifically, we extracted one-word English entries that were represented as a single Japanese word. We then excluded the 1,347 loanwords obtained above from the word pairs, which resulted in 41,366 seed word pairs.

4.2 Evaluation through Correlation

To see if the differences in word embeddings are related to the meaning changes of loanwords, we calculate an evaluation measure indicating the global trend. We first extracted one-to-one translation sentence pairs from Japanese-English News Article Alignment Data (JENAAD) (Utiyama and Isahara, 2003). We then use this set of sentence pairs to calculate the Dice coefficient for each pair of a loanword w_{jpn} and its original English word w_{eng} , which is defined as

$$\frac{2 \times P(w_{\text{jpn}}, w_{\text{eng}})}{P(w_{\text{jpn}}) + P(w_{\text{eng}})}, \quad (2)$$

⁵<https://dumps.wikimedia.org/enwiki/>
<https://dumps.wikimedia.org/jawiki/>

⁶<https://github.com/yohasebe/wp2txt>

⁷<https://en.wikipedia.org/wiki/Help:Infobox>

⁸<http://taku910.github.io/mecab/>

⁹<https://github.com/neologd/mecab-ipadic-neologd>

¹⁰<https://ja.wiktionary.org/wiki/%E3%82%AB%E3%83%86%E3%82%B4%E3%83%AA%3A%E6%97%A5%E6%9C%AC%E8%AA%9E.%E5%A4%96%E6%9D%A5%E8%AA%9E>

¹¹Some of these loanwords may have been introduced into Japanese via other languages. However, in this paper, we regard them as from English as long as they are also used in English.

¹²<http://www.edrdg.org/jmdict/edict.html>

dimension		correlation coefficient	
dim _{jpn}	dim _{eng}	Pearson	Spearman
100	100	0.363	0.443
200	100	0.386	0.471
200	200	0.402	0.474
400	200	0.404	0.487
300	300	0.422	0.492
600	300	0.432	0.506

Table 1: Correlation coefficients between the Dice coefficient and the cosine similarity. dim_{jpn} and dim_{eng} are respectively the dimensions of the Japanese and English word embeddings; i.e., the dim_{jpn} -dimensional space is mapped to the dim_{eng} -dimensional space. All the coefficients are statistically significant (significance level 0.01).

where $P(w_{\text{jpn}}, w_{\text{eng}})$ is the probability that this word pair appears in the same sentence pair, and $P(w_{\text{jpn}})$ and $P(w_{\text{eng}})$ are the generative probabilities of w_{jpn} and w_{eng} . All the probabilities were obtained using the maximum likelihood estimation. The Dice coefficient is a measure of cooccurrence and can be used to extract translate equivalents (Smadja et al., 1996). If the Dice coefficient of a word pair is low, the words in the pair are unlikely to be translation equivalents of each other. Therefore, if the meaning of a loanword has changed from the original English word, its Dice coefficient should be low. In other words, the cosine similarity should be correlated to the Dice coefficient if the cosine similarity is a good indicator of meaning change. We thus calculate the Pearson’s correlation coefficient between the two. In addition, we calculate the Spearman’s rank-order correlation coefficient to examine the relation of the orders given by the Dice coefficient and the cosine similarity.

Note that although we use a parallel corpus for evaluation, it does not mean that we can simply use a parallel corpus for finding meaning changes in loanwords. Parallel corpora are usually much smaller than monolingual corpora and can cover only a small portion of the entire set of loanwords. With the model described in Section 3, we will be able to find meaning changes in loanwords that do not appear in a parallel corpus.

The results for different Japanese and English dimensions, dim_{jpn} and dim_{eng} , are shown in Table 1. Pearson’s correlation coefficients suggest that the cosine similarity is moderately cor-

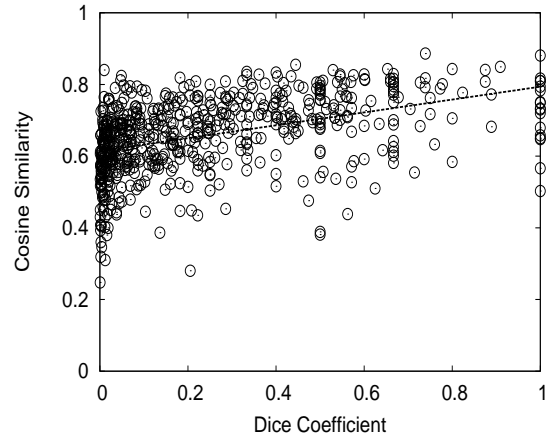


Figure 1: Dice coefficient vs. cosine similarity. Dice coefficients are extracted from a parallel corpus. Cosine similarities are for the embedding vectors of the Japanese loanwords and their English counterparts. The line in the figure is obtained by linear regression.

related with the Dice coefficient except for the case $\text{dim}_{\text{eng}}=100$, which shows weak correlation. Spearman’s rank-order correlation coefficients also suggest that these two are moderately correlated with each other. The result depends on the dimensions of the word embeddings. Basically, larger dimensions tend to have higher correlation coefficients. In addition, when the dimension is decreased (e.g., $\text{dim}_{\text{jpn}} = 600$ to $\text{dim}_{\text{eng}} = 300$), the correlation coefficients tend to be higher, compared with the case where the dimension remains the same (e.g., $\text{dim}_{\text{jpn}} = 300$ to $\text{dim}_{\text{eng}} = 300$). This result is consistent with the report by Mikolov et al. (2013a) that *the word vectors trained on the source language should be several times (around 2x-4x) larger than the word vectors trained on the target language*.

To examine the relation between the Dice coefficient and the cosine similarity in more detail, we plot these values for the bottom row in Table 1, i.e., where the dimensions for Japanese number 600 and the dimensions for English number 300. The scatter plot that we obtained is shown in Figure 1. The line obtained by linear regression is also drawn in the figure.

4.3 Detailed Evaluation on Known Change

Here, we conduct a detailed evaluation on meaning changes that are already known. We selected the ten Japanese loanwords shown in Ta-

w_{eng}	w_{jpn}	w_a	w_b	$\cos(w_{\text{eng}}, w_a) - \cos(w_{\text{jpn}}, w_a)$	$\cos(w_{\text{jpn}}, w_b) - \cos(w_{\text{eng}}, w_b)$
image	imēji	photo	impression	0.097	0.274
corner	cōnā	crossroad	section	0.099	0.115
digest	daijesuto	dissolve	summary	0.047	0.291
bug	bagu	insect	glitch	0.092	0.200
idol	aidoru	deity	popstar	0.127	0.086
icon	aikon	deity	illustration	-0.035	0.145
cunning	kanningu	shrewd	cheating	0.259	0.273
pension	penshon	annuity	hotel	0.368	0.445
nature	neichā	characteristics	magazine	0.106	0.202
driver	doraibā	chauffeur	screwdriver	-0.063	0.158

Table 2: Differences in cosine similarity. The Japanese loanword from *corner* can mean a small section in a larger building or space. The Japanese loanword from *bug* usually means a bug in a computer program. The Japanese loanword from *cunning* usually means cheating on an exam. The Japanese loanword from *nature* is often used to indicate the scientific journal *Nature*. The Japanese loanword from *driver* can mean both a vehicle driver and a screwdriver (the latter meaning was not one of the original word).

ble 2 that are supposed to have different meanings compared with the original English words. Some of these words were taken from a book about loanwords written by Kojima (1988). The others were collected by the authors. We also added two *pivot* words w_a and w_b for each word¹³. For the first nine words, the meaning of pivot word w_a is supposed to be closer to the English word w_{eng} than to the Japanese loanword w_{jpn} , and the meaning of the pivot word w_b is supposed to be closer to the Japanese loanword w_{jpn} than to the English word w_{eng} . It is thus expected that $\cos(w_{\text{eng}}, w_a) - \cos(w_{\text{jpn}}, w_a) > 0$ and $\cos(w_{\text{jpn}}, w_b) - \cos(w_{\text{eng}}, w_b) > 0$ hold true. The last Japanese loanword in Table 2 is used as both pivot words w_a and w_b , but the original English word is not used as w_b . It is thus expected that $\cos(w_{\text{jpn}}, w_b) - \cos(w_{\text{eng}}, w_b) > 0$ holds true, but $\cos(w_{\text{eng}}, w_a) - \cos(w_{\text{jpn}}, w_a) > 0$ might not be necessarily true. The differences in cosine similarities are shown in Table 2. As expected, almost all the differences are positive, which suggests that the difference of the word embeddings captures the meaning change. However, there was one exception:

$$\cos(\text{icon}, \text{deity}) - \cos(\text{icon}_{\text{jpn}}, \text{deity}) = -0.035.$$

¹³Pivot words are not necessarily synonyms of the corresponding English words. They are the words that we think are useful for capturing how the meanings of the loanwords and the original English are different. We also made sure that pivot words themselves are unambiguous.

The cosine similarity between *icon* and *deity* was 0.266, which is smaller than expected. We randomly sampled 100 lines containing *icon* from English Wikipedia text, which we used for calculating word embeddings, and found that the dominant sense of *icon* in Wikipedia is not *a religious painting or figure*, but *a representative person or thing*’ as in the Wikipedia page of a football superstar David Beckham¹⁴:

Beckham became known as a fashion icon, and together with Victoria, the couple became ...

Thus, the reason of *icon*’s anomalous behavior is that the distribution over senses in Wikipedia was a lot different from the expected one.

4.4 Nearest Neighbors

We show in Table 3 the English nearest neighbors of the English word w_{eng} and the Japanese loanword w_{jpn} in the 300-dimensional space of English. Japanese loanwords are mapped from the 600-dimensional space of Japanese into the 300-dimensional space of English. The English word *image* is close in meaning to the word *picture*, as suggested by *jpeg* and *close-up*, while its loanword seems to have a more abstract meaning such as *idealizing*. The nearest neighbors of the English word *digest* are influenced by an American fam-

¹⁴https://en.wikipedia.org/wiki/David_Beckham

w_{eng}	nearest neighbor of w_{eng}		nearest neighbors of w_{jpn}	
image	file	0.774	idealizing	0.671
	jpeg	0.748	stylization	0.665
	jpg	0.724	inescapably	0.665
	closeup	0.694	evoking	0.664
	close-up	0.658	englishness	0.664
corner	corners	0.727	recapped	0.666
	tiltons	0.646	cliff-hanger	0.644
	goerkes	0.643	“blank”	0.642
	uphams	0.629	announcer’s	0.641
	intersection’s	0.627	sports-themed	0.632
digest	digests	0.609	recaps	0.717
	digest’s	0.594	wrap-up	0.697
	reader’s	0.591	recapped	0.695
	wallace-reader’s	0.573	preview	0.693
	wallace/reader’s	0.556	recap	0.690
bug	bugs	0.672	heartbleed	0.714
	leaf-footed	0.605	workaround	0.695
	motherhead	0.590	workarounds	0.686
	harpactorinae	0.582	glitches	0.684
	thread-legged	0.579	copy-on-write	0.684
icon	icons	0.750	swoosh	0.701
	iverskaya	0.580	viewport	0.694
	nicopeia	0.579	crosshair	0.691
	eleusa	0.570	upper-left	0.684
	derzhavnaya	0.569	wireframe	0.680
nature	teiči	0.649	phytogeography	0.684
	søraust-svalbard	0.643	ethological	0.679
	naturans	0.627	life-history	0.676
	naturata	0.623	paleoclimatology	0.671
	naturing	0.623	archaeoastronomy	0.670
driver	drivers	0.837	driver	0.762
	driver’s	0.703	race-car	0.689
	car	0.685	mechanic	0.649
	co-drivers	0.655	harvick’s	0.645
	owner-driver	0.653	andretti’s	0.642

Table 3: English words that are nearest w_{eng} and w_{jpn} . w_{jpn} is a Japanese loanword and w_{eng} is the original English word. w_{jpn} is mapped into the English vector space. Only words that appear more than 100 times in the Wikipedia corpus are considered as candidates of the nearest neighbors. The value next to each word is the cosine similarity between the nearest neighbor word and w_{eng} or w_{jpn} .

ily magazine *Reader's Digest*¹⁵ by Wallaces, but the terms related to *summary* do not appear in the top-5 list, except for *digest* itself. In contrast, its loanword seems to mean *wrap-up*. We now return to the English word *icon* that was mentioned as an exception in Section 4.3. Besides *icons*, the nearest neighbors of *icon* are *iverskaya*, *nicopeia*, *eleusa*, and *derzhavnaya*. These four words are all related to religious paintings or figures, but they have low cosine similarities. The other parts of the table are also mostly interpretable. The nearest neighbors of w_{eng} *nature* look uninterpretable at a first glance, but they are influenced by the Sjøraust-Svalbard Nature Reserve in Norway, and *Natura naturans*, which is a term associated with the philosophy of Baruch Spinoza.

4.5 Ranking of Word Pairs According to Similarity

Here, we investigate the possibility of whether the similarity calculated in the mapped space can be used to detect the loanwords that are very different from or close to the original English words. We show the 20 words with the lowest cosine similarities and the 20 words with highest cosine similarities in Table 4. First, let us take a look at the words on the right, which have high similarities. Most of them are technical terms (e.g., *hexadecane* and *propylene*), and domain-specific terms such as musical instruments (e.g., *piano* and *violin*) and computer-related terms (*computer* and *software*). This result is consistent with the fact that the meanings of technical terms tend not to change, at least for Japanese (Nishiyama, 1995). Next, let us take a look at the words on the left, which have low similarities. Many of them are actually ambiguous, and this ambiguity is often due to the Japanese phonetic system. For example, *lighter* and *writer* are assigned to the same loanword in Japanese, because the Japanese language does not distinguish the consonants *l* and *r*. The words *clause*, *close* and *clothe* are also assigned to the same loanword also because of the Japanese phonetic system. Other words are used as parts of named entities, also resulting in low similarity. For example, the Japanese loanword for *refer* is more often used as *Rifaa*, the name of a city in Bahrain, but hardly as *refer*. The loanword for *irregular* is often used as part of a video game title *Irregular Hunter*. However, we can also find words with sig-

¹⁵<http://rd.com>

dissimilar pair		similar pair	
w_{eng}	cosine	w_{eng}	cosine
lac	0.225	piano	0.886
refer	0.245	violin	0.881
police	0.247	cello	0.881
spread	0.251	hexadecane	0.864
mof	0.261	propylene	0.857
pond	0.270	keyboard	0.855
inn	0.274	clarinet	0.851
ism	0.279	cheese	0.849
lighter	0.280	mayonnaise	0.848
root	0.281	software	0.847
tabu	0.284	methanol	0.843
gnu	0.293	hotel	0.843
thyme	0.296	chocolate	0.841
clause	0.310	computer	0.840
board	0.315	engine	0.840
present	0.319	globalization	0.835
coordinate	0.337	tomato	0.833
expanded	0.341	trombone	0.832
irregular	0.342	recipe	0.831
measure	0.346	antimony	0.829

Table 4: Twenty words with the lowest similarities and twenty words with the highest similarities.

nificant changes in meaning, such as *present*¹⁶ and *coordinate*¹⁷. Therefore, the result suggests that the similarity calculated by our method has the capability of detecting changes in the meanings of loanwords, but we need to filter out the words that are ambiguous in the Japanese phonetic system.

We manually evaluated the 100 words that have the lowest similarities to the corresponding loanwords including the 20 words shown in Table 4. Among the 100 words, 21 words are influenced by ambiguity, and 19 are influenced by named entities. Among the remaining 60 words, 57 are judged to be semantically different from their loanwords. For the other three words, the embeddings would not be quite accurate probably due to their infrequency in either the English or the Japanese corpora used for training.

4.6 Evaluation for Educational Use

To see if the obtained word embeddings of English and Japanese can assist in language learn-

¹⁶In Japanese, *present* usually means a gift, or to give a gift, but hardly to show or introduce.

¹⁷In Japanese, this word usually means to match one's clothes attractively.

ing, for purposes such as lexical-choice error correction, we evaluate their usefulness by using the writings of Japanese learners of English. Specifically, we use the Lang-8 English data set (Mizumoto et al., 2011)¹⁸ to calculate the Dice coefficient instead of JENAAD. This dataset consists of sentences originally written by learners, some of which have been corrected by (presumably) native speakers of English. Because we target embeddings of English and Japanese, we only use English sentences written by Japanese among other learners of English. Of those, approximately one million sentences have corresponding corrections. With these sentence pairs, we calculate the Dice coefficient just as in Section 4.2. The coefficient measures how often a word co-occurs in the original sentences and corresponding corrections. If a word is often corrected to another, it tends to appear only in the original sentences and not in the corresponding corrections, and thus, its Dice coefficient becomes small, and vice versa. In other words, the Dice coefficient roughly measures how often a word is corrected in the Lang-8 English data. Considering this, we compare the cosine similarity based on the proposed method with the Dice coefficient by means of the Pearson's correlation to evaluate how effective the cosine similarity is in predicting words in which lexical-choice errors likely occur¹⁹; the higher the correlation is, the better the cosine similarity is as an indicator of lexical-choice errors. Note that we apply lemmatization to all words both in the original sentences and in the corresponding corrections when calculating the Dice coefficient in order to focus only on lexical-choice errors²⁰.

It turns out that the value of the Pearson's correlation coefficient shows a milder correlation ($\rho=0.302$; significant at the significance level $\alpha=0.01$) in this dataset than in JENAAD. Some loanwords having the almost equivalent senses in English have high values both for the cosine similarity and the Dice coefficient; examples are musical instruments

¹⁸<http://cl.naist.jp/nldata/lang-8/>

¹⁹Some of the words in the loanword list are too infrequent to calculate the Dice coefficient in the Lang-8 data set. Accordingly, we excluded those appearing fewer than 30 times in it when calculating the Pearson's correlation.

²⁰Other grammatical errors including errors in number and inflection often appear in the Lang-8 English data, which are mistakenly included in lexical-choice errors in the calculation of the Dice coefficient. Lemmatization reduces their influences to some extent.

such as *piano* (cos=0.886, Dice=0.951) and *violin* (cos=0.881, Dice=0.914); computer terms *computer* (cos=0.840, Dice=0.865) and *software* (cos=0.847, Dice=0.880) as has discussed in Section 4.5. Moreover, some that do not have equivalent senses show mild correspondences (e.g., *sentence* (cos=0.493, Dice=0.346); *note* (cos=0.470, Dice=0.352)).

By contrast, most of the others show less correspondence. One possible reason is that in the Lang-8 English data, corrections are applied to grammatical errors other than lexical choices, which undesirably decreases the Dice coefficient. Typical examples are errors in number (singular countable nouns are often corrected as corresponding plural nouns; e.g., *book* → *books*) and in inflection (e.g., *book* → *booked*). Therefore, loanwords whose corresponding English words undergo word-form changes less often tend to show strong correspondences as can be seen in the above examples (i.e., *software* and *piano*). This can be regarded as noise in the use of the Lang-8 data set. As mentioned above, we applied lemmatization to reduce the influences by noise. More sophisticated methods such as word alignment might improve the accuracy of the evaluation.

5 Conclusions

We computationally analyzed semantic changes of Japanese loanwords. We used the word embeddings of Japanese and English, and mapped the Japanese embeddings to the space of English, where we calculated the cosine similarity of a Japanese loanword and its original English word. We regarded this value as an indicator of semantic change. We evaluated our methodology in a number of ways.

To detect semantic changes accurately, we have to filter out the words that are ambiguous in the Japanese phonetic system. Such words tend to have low cosine similarities. One direction for future work is application of the technique to similar tasks. For example, we can use our method to analyze semantic changes of cognates. There are also a number of ways to investigate semantic changes of loanwords in more detail. For example, we can examine the relation between the semantic change of a loanword and the time at which the word was introduced in the target language. Hamilton et al. (2016) reported that they

used word embeddings to show the relation between semantic changes and polysemy. It would be interesting to see if similar results are obtained for loanwords.

Acknowledgments

This work was partly supported by Grant-in-Aid for Young Scientists (B) Grant Number JP26750091. Yoshifumi Kawasaki is supported by JSPS KAKENHI Grant Number 15J04335.

References

- David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834.
- Keith Barrs. 2013. Assimilation of English vocabulary into the Japanese language. *Studies in Linguistics and Language Teaching*, 24:1–12.
- Alan D. Blair and John Ingram. 1998. Loanword formation: a neural network approach. In *Proceedings of SIGPHON Workshop on the Computation of Phonological Constraints*, pages 45–54.
- James Breen, Timothy Baldwin, and Francis Bond. 2012. Segmentation and translation of Japanese multi-word loanwords. In *Proceedings of Australasian Language Technology Association Workshop*, pages 61–69.
- Jim W. Breen. 2000. A WWW Japanese dictionary. *Japanese Studies*, pages 313–317.
- Frank E. Daulton. 2009. A sociolinguist explanation of Japan’s prolific borrowing of English. *The Ryukokou Journal of Humanities and Sciences*, pages 29–36.
- Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the European Chapter of Association for Computational Linguistics*, pages 462–471.
- Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 748–756.
- William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL2016)*, pages 1489–1501. Association for Computational Linguistics.
- Yoichiro Hasebe. 2006. Method for using Wikipedia as Japanese corpus (in Japanese). *Doshisha studies in language and culture*, 9(2):373–403.
- Mark Irwin. 2011. *Loanwords in Japanese*. John Benjamins Publishing Company.
- Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 229–238. IEEE Press.
- Gillian Kay. 1995. English loanwords in Japanese. *World Englishes*, pages 67–76.
- Yoshiro Kojima, editor. 1988. *Nihongo no imi eigo no imi (meanings in Japanese, meanings in English (translated by the authors of this paper))*. Nan’undo.
- Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 230–237.
- Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2014. Statistically significant detection of linguistic change. In *Proceedings of the 24th World Wide Web Conference (WWW)*, pages 625–635.
- Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2016. Freshman or fresher? quantifying the geographic variation of language in online social media. In *Proceedings of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016)*, pages 615–618.
- Lingshuang Jack Mao and Mans Hulden. 2016. How regular is Japanese loanword adaptation? A computational study. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 847–856.
- Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.
- Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

- Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 147–155.
- Sen Nishiyama. 1995. Speaking English with a Japanese mind. *World Englishes*, pages 1–6.
- Frank Smadja, Kathleen R. McKeown, and Vasileios Hatzivassiloglou. 1996. Translating collocations for bilingual lexicons: a statistical approach. *Computational Linguistics*, pages 1–38.
- Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning Japanese-English news articles and sentences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 72–79.
- Will Zou, Richard Socher, Daniel Cer, and Christopher Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1393–1398.

Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists

Gerhard Jäger
Tübingen University
Institute of Linguistics
Tübingen, Germany
gerhard.jaeger@
uni-tuebingen.de

Johann-Mattis List
CRLAO / AIRE
EHESS / UPMC
PARIS
info@lingpy.org

Pavel Sofroniev
Tübingen University
Institute of Linguistics
Tübingen, Germany
pavel.sofroniev@
student.uni-tuebingen.de

Abstract

Most current approaches in phylogenetic linguistics require as input multilingual word lists partitioned into sets of etymologically related words (cognates). Cognate identification is so far done manually by experts, which is time consuming and as of yet only available for a small number of well-studied language families. Automatizing this step will greatly expand the empirical scope of phylogenetic methods in linguistics, as raw wordlists (in phonetic transcription) are much easier to obtain than wordlists in which cognate words have been fully identified and annotated, even for under-studied languages. A couple of different methods have been proposed in the past, but they are either disappointing regarding their performance or not applicable to larger datasets. Here we present a new approach that uses support vector machines to unify different state-of-the-art methods for phonetic alignment and cognate detection within a single framework. Training and evaluating these method on a typologically broad collection of gold-standard data shows it to be superior to the existing state of the art.

1 Introduction

Computational historical linguistics is a relatively young sub-discipline of computational linguistics which uses computational methods to uncover how the world's 7 000 human languages have developed into their current shape. The discipline has made great strides in recent years. Exciting progress has been made with regard to automated language classification (Bowerman and Atkin-

son, 2012; Jäger, 2015), inference regarding the time depth and geographic location of ancestral language stages (Bouckaert et al., 2012), or the identification of sound shifts and the reconstruction of ancestral word forms (Bouchard-Côté et al., 2013), to mention just a few. Most of the mentioned and related work relies on multilingual word lists manually annotated for *cognacy*. Unlike the classical NLP conception, cognate words are here understood as words in different languages which are *etymologically related*, that means, they have regularly developed from a common ancestral form, such as both English *tooth* and German *Zahn* 'tooth' that go back to an earlier Proto-Germanic word *tanθ-* with the same meaning. Manual cognate classification is a slow and labor intensive task requiring expertise in historical linguistics and intimate knowledge of the language family under investigation. From a methodological perspective, it can further be problematic to build phylogenetic inference on expert judgments, as the expert annotators necessarily base their judgments on certain hypotheses regarding the internal structure of the language family in question. In this way, the human-annotated cognate sets bear the danger of circularity. Deploying automatically inferred cognate classes thus has two advantages: it avoids the bias inherent in manually collected expert judgments and it is applicable to both well-studied and under-studied language families.

In the typical scenario, the researcher has obtained a collection of multilingual word lists in phonetic transcription (e.g. from field research or from dictionaries) and wants to classify them according to cognacy. Such datasets usually cover many languages and/or dialects (from scores to hundreds or even thousands) but only a small number of concepts (often the 200-item or 100-item Swadesh list or subsets thereof). The machine

learning task is to perform cross-linguistic clustering. There exists a growing body of gold standard data, i.e. multilingual word lists covering between 40 and 210 concepts which are manually annotated for cognacy (see Methods section for details). This suggests a supervised learning approach. The challenge here is quite different from most machine learning problems in NLP though since the goal is not to identify and deploy language-specific features based on a large amount of mono- or bi-lingual resources. Rather, the gold standard data have to be used to find cross-linguistically informative features that generalize across arbitrary language families. In the remainder of this paper we will propose such an approach, drawing on and expanding related work such as List (2014b) and Jäger and Sofroniev (2016).

2 Previous Work

Cognate detection is a *partitioning task*: a *clustering task* which does not necessarily assume a hierarchy. An early approach (Dolgopolsky, 1964) is based on the idea of *sound classes*: In order to reduce the phonetic space and to guarantee comparability across languages, sounds are clustered into classes which frequently occur in correspondence relation in genetically related languages. Dolgopolsky proposed a very rough sound class system, proposing to group all consonants into ten classes ignoring vowels. When converting all transcriptions in the data to their respective sound classes, one can use different criteria to assign words resembling each other in their sound classes to the same set of cognate words. Turchin et al. (2010) further formalized this approach and employed a modified sound class schema of 9 vowel classes to test the Altaic hypothesis. Their *Consonant Class Matching* (CCM) approach was reported to produce a low rate of false positives. Unfortunately, the rate of false negatives is also very high (List, 2014b). This is especially due to the lack of flexibility of the procedure, which hard-codes sounds to classes, ignoring that sound change is usually based on fine-grained transitions.

An alternative family of approaches to cognate detection circumvents this problem by first calculating distances or similarities between pairs of words in the data, and then feeding those scores to a flat clustering algorithm which partitions the words into cognate sets. This workflow

is very common in evolutionary biology, where it is used to detect homologous genes and proteins (Bernardes et al., 2015). Two basic families of partitioning algorithms can be distinguished: hierarchical cluster algorithms and graph-based algorithms. Hierarchical cluster algorithms are based on classical agglomerative cluster algorithms (Sokal and Michener, 1958), but terminate when a user-defined threshold of average similarities among clusters is reached. In graph-based partitioning algorithms (Andreopoulos et al., 2009), words are represented as nodes in a network and links between nodes represent similarities. When clustering, links are added and removed until the nodes are partitioned into homogeneous groups (van Dongen, 2000).

More important than the clustering algorithm one uses is the computation of pairwise similarity scores between words. Here, different measures have been tested, ranging from simple string distance metrics (Bergsma and Kondrak, 2007), via enhanced sound-class-based alignment algorithms (SCA, List 2014a), to iterative frameworks in which segmental similarities between sounds are either iteratively inferred from the data (Steiner et al., 2011), or aggregated using machine learning techniques (Hauer and Kondrak, 2011). Frameworks may differ greatly regarding their underlying workflow. While the LexStat algorithm by List (2014b) uses a permutation method to compute individual segmental similarities between individual language pairs which are then fed to an alignment algorithm, the *PMI similarity approach* by Jäger (2013) infers general segmental similarities between sounds from an exhaustive parameter training procedure.

3 Materials

Benchmark data for training and testing was assembled from different previous studies and considerably enhanced by unifying semantic and phonetic representations and correcting numerous errors in the datasets. Our collection was taken from six major sources (Greenhill et al., 2008; Dunn, 2012; Wichmann and Holman, 2013; List, 2014b; List et al., 2016b; Menecier et al., 2016)¹ and

¹The Indo-European data from `ielex.mpi.nl` were accessed on 4-26-2016. The Austronesian data from the *Austronesian Basic Vocabulary Database* (ABVD, `language.psy.auckland.ac.nz/austronesian/`) were accessed on 12-2-2015. Among the 395 languages covered by ABVD, we only used a randomly selected subset of 100 lan-

Dataset	Words	Conc.	Lang.	Families	Cog.	Div.
ABVD (Greenhill et al. 2008)	12414	210	100	Austronesian	3558	0.27
Afrasian (Militarev 2000)	790	40	21	Afro-Asiatic	355	0.42
Bai (Wang 2006)	1028	110	9	Sino-Tibetan	285	0.19
Chinese (Hu 2004)	2789	140	15	Sino-Tibetan	1189	0.40
Chinese (Bijng Dxu 1964)	3632	179	18	Sino-Tibetan	1225	0.30
Huon (McElhanon 1967)	1176	84	14	Trans-New Guinea	537	0.41
IELex (Dunn 2012)	11479	208	52	Indo-European	2459	0.20
Japanese (Hattori 1973)	1983	199	10	Japonic	456	0.15
Kadai (Peiros 1998)	400	40	12	Tai-Kadai	103	0.17
Kamasau (Sanders 1980)	271	36	8	Torricelli	60	0.10
Lolo-Burmese (Peiros 1998)	570	40	15	Sino-Tibetan	101	0.12
Central Asian (Manni et al. 2016)	15903	183	88	Altaic (Turkic), Indo-European	895	0.05
Mayan (Brown 2008)	2841	100	30	Mayan	844	0.27
Miao-Yao (Peiros 1998)	208	36	6	Hmong-Mien	70	0.20
Mixe-Zoque (Cysouw et al. 2006)	961	100	10	Mixe-Zoque	300	0.23
Mon-Khmer (Peiros 1998)	1424	100	16	Austroasiatic	719	0.47
ObUgrian (Zhvlov 2011)	2006	110	21	Uralic	229	0.06
Tujia (Starostin 2013)	498	107	5	Sino-Tibetan	164	0.15

Table 1: Benchmark data used for the study. Items on red background were used for testing, and the rest for training. Items on white are available in ASJP transcription; all others are available in IPA transcription. The last column lists the diversity of each dataset by dividing the number of actual cognates by the number of potentially different cognates (List 2014:188).

covers datasets ranging between 100 and 210 concepts translated into 5 to 100 languages from 13 different language families.

Modifications introduced in the process of preparing the datasets included (a) the correction of errata (e.g. orthographic forms in place of phonetic representations), (b) the replacement of non-IPA symbols with their IPA counterparts (e.g. $\text{t} \rightarrow \text{t}$ or $' \rightarrow ?$), (c) the removal of non-IPA symbols used to convey meta-information (e.g. $\%$), (d) removal of extraneous phonetic representation variants, and (e) the removal of morphological markers. In addition, all concept labels in the different datasets were linked to the Concepticon (<http://concepticon.clld.org>, List et al. 2016a), a resource which links concept labels

language since the computational effort would have been impractical otherwise. For all data sets, only entries containing both a phonetic transcription and the cognate classification were used.

language	iso	gloss	gloss_id	transcr.	cogn._class
ELFDALIAN	qov	woman	962	'kɛlɪŋg	woman:Ag
DUTCH	nld	woman	962	vrcu	woman:B
GERMAN	deu	woman	962	fraü	woman:B
DANISH	dan	woman	962	'g ^h venə	woman:D
DANISH_FJOLDE		woman	962	kvin'	woman:D
DANISH_LAU		woman	962	'kvim; folk	woman:D
LATIN	lat	woman	962	'mulier	woman:E
LATIN	lat	woman	962	'femina	woman:G
ENGLISH	eng	woman	962	womən	woman:H
GERMAN	deu	woman	962	vaip	woman:H
DANISH	dan	woman	962	'ðemə	woman:K

Table 2: Sample entries for *woman* in IELex. The cognate class identifier in the last column consists of a the concept label and an arbitrary letter combination. If two words share the same cognate class identifier, they are marked as cognate.

to standardized concept sets in order to ease the exchange and standardization of cross-linguistic datasets. A small sample of the entries extracted from the IELex data is shown in Table 2 for illustration.

4 Methods

Unlike many other supervised or semi-supervised clustering tasks, the set of cluster labels to be inferred is disjoint from the gold standard labels. Therefore we chose a two-step procedure: (1) A similarity score for each pair of synonymous words from the same dataset is inferred using supervised learning, and (2) these inferred similarities are used as input for unsupervised clustering.

As for subtask (1), the relevant gold standard information are the labels “cognate” and “not cognate” for pairs of synonymous words. The sub-goal is to predict a probability distribution over these labels for unseen pairs of synonymous words. This is achieved by training a Support Vector Machine (SVM), followed by Platt scaling (Platt, 1999). The SVM primarily operates on two string similarity measure from the literature, *PMI similarity* Jäger (2013) and *LexStat similarity* (List, 2014b), which are both known to generalize well across languages and language families. We also used some auxiliary features from (Jäger and Sofroniev, 2016), which are derived from string similarities. For the clustering subtask (2), we followed List et al. (2016b) and List et al. (2017) in using the Infomap algorithm (Rosvall and Bergstrom, 2008).

The gold standard data were split into a training set and a test set. Feature selection for subtask (1) and parameter training for subtask (2) were achieved via cross-validation over the train-

ing data. For evaluation, we trained an SVM on all training data and used it to perform automatic clustering on the test data.

The remainder of this section spells out these steps in detail.

4.1 String Similarity Measures

Our strategy is to first calculate string similarities and distances between pairs of words denoting the same concept and then inferring a partition of the corresponding words from those similarities or distances via a partitioning algorithm. For word comparison we utilize two recently proposed string similarity measures.

The first string similarity measure is the one underlying the above-mentioned LexStat algorithm for automatic cognate detection (List, 2014b). The core features of the string similarity produced by the LexStat algorithm include (a) an enhanced sound-class model of 28 symbols, including tone symbols for the handling of South-East Asian tone languages, (b) a linguistically informed scoring function derived from frequently recurring directional sound change processes, and (c) a prosodic tier which automatically defines a prosodic context for each sound in a word and thus allows for a rough handling of context. The LexStat algorithm for determining string similarities can be roughly divided into four stages. In a first stage, words for the same concept in each language pair are aligned, using the SCA algorithm for phonetic alignment (List, 2014b), both globally and locally, and correspondences in the word pairs with a promising score are retained. At the same time, a randomized distribution of expected sound correspondences is calculated, using a permutation method (Kessler, 2001) in which the wordlist are shuffled, so that words denoting different concepts, which are much more likely to be not cognate, are aligned instead. In a second step, both distributions are compared, and log-odds scores (Durbin et al., 2002) for each segment pair $s_{x,y}$ are calculated (List, 2014b, 181). In a third step, the new scoring function is used to re-align the words, using a semi-global alignment algorithm which ignores prefixes or suffixes occurring in one of two strings (Durbin et al., 2002), and the similarity scores produced by classical alignment algorithms are normalized to similarity scores using the formula by Downey et al. (2008)

$$D = \frac{2 \cdot S_{AB}}{S_A + S_B} \quad (1)$$

where S_{AB} is the similarity score of an alignment of two words A and B produced by the SCA method, and S_A and S_B are the similarity scores produced by the alignment of A and B with themselves.²

In Jäger (2013) a data-driven method for determining string similarities is proposed which we will refer to as *PMI similarity*, as it is based on the notion of *Pointwise Mutual Information* between phonetic segments. It has successfully been used for phylogenetic inference in Jäger (2015). The method operates on phonetic strings in ASJP transcription (Brown et al., 2013) without diacritics, i.e., each segment is assigned one out of only 41 sound classes.

The PMI score of two sound classes a, b is defined as

$$\text{PMI}(a, b) \doteq \log \frac{s(a, b)}{q(a)q(b)}, \quad (2)$$

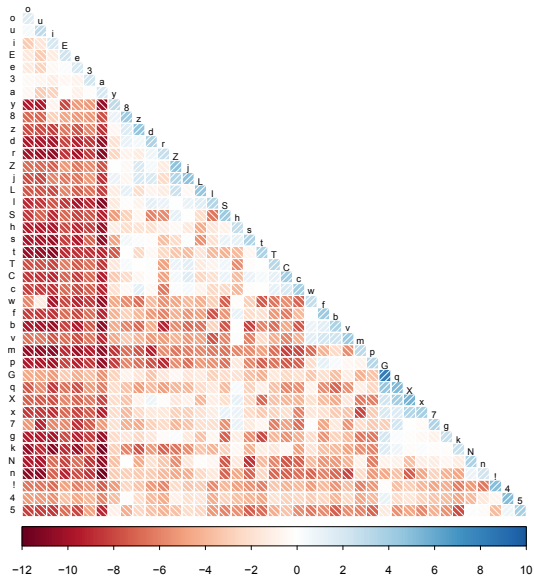
where $s(a, b)$ is the probability of a and b being aligned to each other in a pair of cognate words, and $q(a), q(b)$ are the probabilities of occurrence of a and b respectively. Sound pairs with positive PMI score provide evidence for cognacy, and vice versa.

To estimate the likelihood of sound class alignments, a corpus of *probable cognate pairs* was compiled from the ASJP data base³ using two heuristics. First, a crude similarity measure between wordlists, based on Levenshtein distance, was defined and the 1% of all ASJP doculect⁴ pairs with highest similarity were kept as *probably related*. Second, the normalized *Levenshtein distance* was computed for all translation pairs from probably related doculects. Those with a distance below a certain threshold were considered as *probably cognate*. These probable cognate pairs were used to estimate PMI scores. Subsequently, all translation pairs were aligned via the Needleman-Wunsch algorithm Needleman and Wunsch (1970) using the PMI scores from the previous step as weights. This resulted in a measure of string similarity, and all pairs above a certain similarity

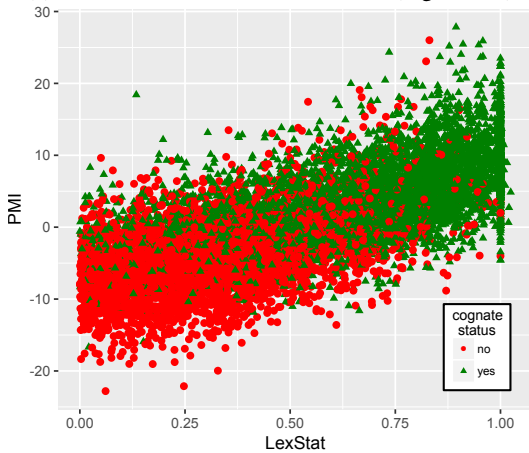
²The original LexStat algorithm uses distance scores by subtracting the similarity score from 1.

³The ASJP database Wichmann et al. (2013), available from <http://asjp.cild.org/>, is a collection of 40-item Swadesh lists from more than 6,000 languages and dialects covering all regions of the globe.

⁴*Doculect* is a neutral term for a linguistic variety which is documented in some coherent way, leaving the issue of distinguishing between languages and dialects aside.



(a) PMI scores between sound classes (Jäger 2015)



(b) Joint distribution of string similarities (IPA training data)

Figure 1: PMI scores (a) and string similarities (b) of cognate and non-cognate word pairs from the training data.

threshold were treated as probable cognates in the next step. This procedure was repeated ten times. In the last step, app. 1.3 million probable cognate pairs were used to estimate the final PMI scores.

The PMI scores thus obtained are visualized in Figure 1a (numerical values are available from the Supplementary Material of Jäger (2015)). The aggregate PMI score of a pair of aligned strings (where gaps may be inserted at any position) is defined as the sum of the PMI scores of the aligned symbol pairs. Matching a symbol with a gap incurs a penalty, with different penalties for initial and non-initial positions in a sequence of consecutive gaps.⁵ The similarity $s(w_1, w_2)$ between two

⁵The values of the gap penalties were taken from Jäger (2013), where the method of estimating them is described.

strings w_1, w_2 is then defined as minimal aggregate PMI score for all possible alignments. It can be computed efficiently via the Needleman-Wunsch algorithm.

There are major conceptual differences on how the two similarity measures are derived. LexStat similarity estimates separate scores between each pair of doculects, thus utilizing *regular sound correspondences*, while PMI similarity uses the same PMI scores regardless of the doculects compared. LexStat alignments further capture a prosodic tier which allows for a rough modeling of phonetic context and reflects theories on the importance of phonetic strength in sound change processes (Geisler, 1992), while the parameters used for computing PMI similarities are estimated in a purely data-driven way without using specifically linguistic insights beyond the classification of sounds into ASJP sound classes. The parameters of the PMI framework are statistically estimated using a large amount (more than 1 000 000 word pairs) of cross-linguistically diverse data. In contrast, LexStat’s initial alignment algorithm is based on manually assigned parameters, and the final parameters are estimated empirically from the word pairs in the doculects being compared, and no external information is being applied. As a result, the algorithm needs a minimum of 100 concepts to yield reliable results and it yields notably better results with more than 200 words (List, 2014b; List, 2014a).

The joint distribution of LexStat and PMI string similarities for cognate and non-cognate pairs within our training set is visualized in Figure 1b.

Despite those differences, the two measures capture a similar signal; for the data from List (2014b) and List et al. (2016b), e.g., their correlation is as high as 0.727. Also, both variables are contain similar information about the binary *cognate/not cognate* variable. Figure 2 shows the *Precision-Recall curves* (cf. for instance Manning and Schütze, 1999) for LexStat and PMI similarity. While the curves are slightly different (LexStat achieves a higher precision for low recalls and PMI for high recalls), the areas under the curve are almost identical (0.893 for LexStat and 0.880 for PMI).

4.2 Workflow

In this study, we utilized both string similarity measures discussed above, as well as a collec-

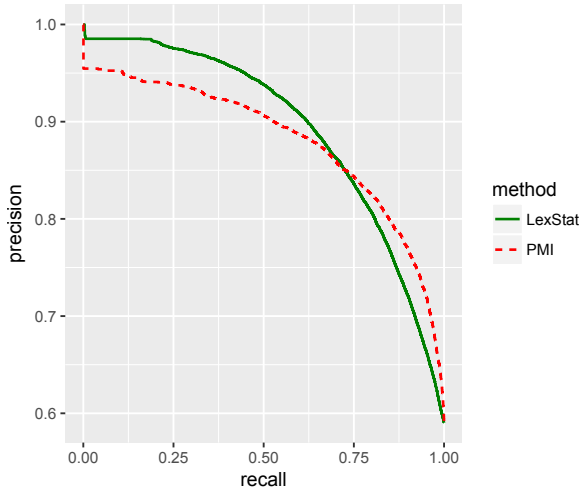


Figure 2: Precision-Recall curves for LexStat and PMI string similarities, based on the evaluation of the word pairs from the data by List (2014a).

tion of auxiliary predictors pertaining to the similarity of the doculects compared and the differential diachronic stability of lexical meanings, to infer cognate classifications. We chose a supervised learning approach using a Support Vector Machine (SVM) for this purpose. The overall workflow is shown in Figure 3. It consists of two major parts. During the first phase (the upper part in the figure shown in red), a SVM is trained on a set of training data and then used to predict the *probability of cognacy* between pairs of words from a set of test data. During the second phase (lower part in the figure, shown in green), those probabilities are used to cluster the words from the test set into *inferred cognacy classes*. The system is evaluated by comparing the inferred classification with the expert classification. We used the three largest data sets at our disposal (cf. the datasets colored in red in Table 1), *ABVD*, *Central Asian*, and *IELex*, for testing and all other datasets for training.

4.3 Support Vector Machine Training

Each data point during the first phase is a pair of words w_1, w_2 (i.e., a pair of phonetic strings) from doculects L_1, L_2 from data set S , both denoting the same concept c . It is mapped to a vector of values for the following features:⁶

1. LexStat string similarity between w_1 and w_2 (computed with LingPy, List and Forkel,

⁶Features 2–5 are taken from (Jäger and Sofroniev, 2016). The other features used there (calibrated PMI distances and their logarithms, and the logarithm of doculect similarity) did not improve results under cross-validation over the training data.

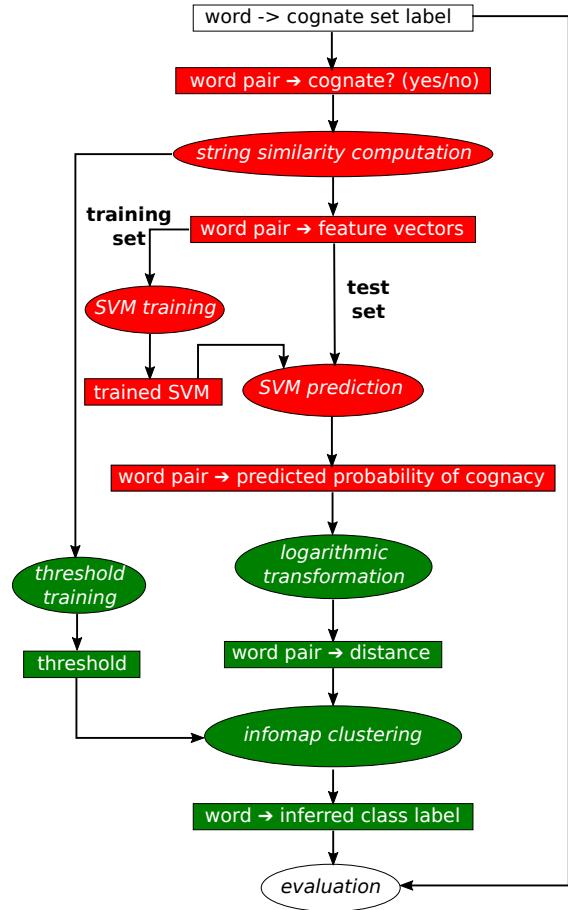


Figure 3: Workflow for supervised learning and prediction. Boxes and ellipses represent data and computations respectively.

2016) ,

2. PMI string similarity between w_1 and w_2 ,
3. doculect similarity between L_1 and L_2 as defined in Jäger (2013),⁷
4. mean word length (measured in number of segments) of words for concepts c within S .
5. correlation coefficient between PMI string similarity and doculect similarity across all word pairs denoting concept c within S .⁸

The marginal distributions for cognate and non-cognate pairs of those features (for the data from List (2014b) and List et al. (2016b)) is displayed in Figure 4. It can be discerned from these plots that word length is a negative predictor and the other four features are positive predictors for cognacy.

⁷We refrain from recapitulating the full definition here for reasons of space. Essentially this amounts to the average PMI similarity between synonymous word pairs from L_1 and L_2 .

⁸The last two features represent measures of the diachronic stability of concepts, based on Dellert and Buch (2016).

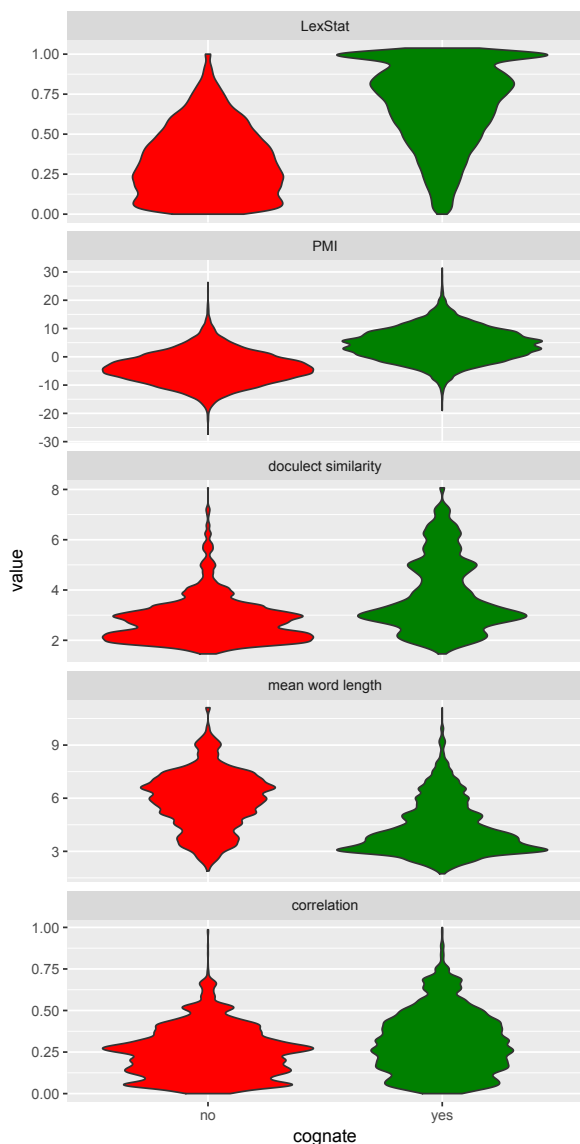


Figure 4: Distribution of features values for cognate and non-cognate word pairs

The fact that word length is a negative predictor of cognacy arguably results from the interplay of two known regularities. (1) Pagel et al. (2007) present evidence that diachronic stability of concepts is positively correlated with their usage frequency in modern corpora. (2) According to *Zipf’s Law of Abbreviation* (Zipf, 1935), there is a negative correlation between the corpus frequency of words and their lengths. Taken together, this means that concepts usually being expressed by short words tend to have a high usage frequency and therefore tend to be diachronically stable. Therefore we expect a higher proportion of cognate pairs among concepts expressed by short words than among those expressed by long words.

As the data points within the training set are mutually non-independent, we randomly chose one word pair per concept and data set for training the SVM. During the training phase, we used cross-validation over the data sets within the training set (i.e., using one training data set for validation and the other training data sets for SVM training) to identify the optimal kernel and its optimal parameters. This was carried out by completing both phases of the work flow and optimizing the *Adjusted Rand Index* (see Subsection 4.5) of the resulting classification. Training and prediction was carried out using the `svm` module from the Python package `sklearn` (<http://scikit-learn.org/stable/modules/svm.html>), which is based on the LIBSVM library (Fan et al., 2005). Predicting class membership probabilities from a trained SVM was carried out using Platt scaling (Platt, 1999) as implemented in `sklearn` (<http://scikit-learn.org>). This results in a *predicted probability of cognacy* $p(w_1, w_2|c, S)$ for each data point. The best cross-validation performance was achieved with a linear kernel with a penalty value of $C = 0.82$. Polynomial and RBF-kernels performed slightly worse. Also, we found that leaving out any subset of the features decreases performance.

4.4 Cognate Set Partitioning

In order to cluster the words into sets of potentially cognate words, we follow recent approaches by List et al. (2016b) and List et al. (2017) in using Infomap (Rosvall and Bergstrom, 2008), an algorithm which was originally designed for the detection of communities in large social networks, to detect “communities” of related words. Infomap uses random walks in undirected networks to identify the best way to assign the nodes in the network, that is, in our case, the words, to distinct groups which form a homogeneous class.

For each data set D and each concept c covered in D , a network was constructed. The vertices are all words from D denoting c . Two vertices are connected if and only if the corresponding words are predicted to be cognate with a probability $\geq \theta$ according to SVM prediction + Platt scaling. The optimal value for θ was determined as 0.66 via cross-validation over the training data. Infomap was then applied to this network, resulting in an assignment of class labels to vertices/words.

data set	Adjusted Rand Index		B-Cubed Precision		B-Cubed Recall		B-Cubed F-Score	
	LexStat	SVM	LexStat	SVM	LexStat	SVM	LexStat	SVM
aggregated	0.676	0.683	0.868	0.847	0.838	0.869	0.850	0.855
Austronesian	0.545	0.588	0.791	0.781	0.801	0.855	0.796	0.817
Central Asian	0.866	0.843	0.916	0.883	0.962	0.981	0.938	0.929
Indo-European	0.618	0.619	0.896	0.877	0.750	0.770	0.817	0.820

Table 3: Evaluation results on the test data for the benchmark method (LexStat) and our method (SVM) according to Adjusted Rand Index and B-Cubed precision, recall, and F-score

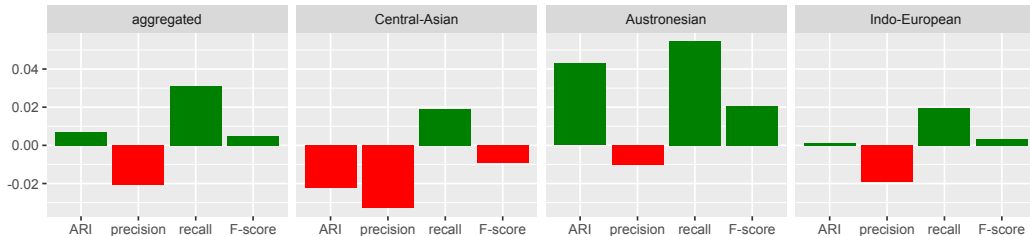


Figure 5: Evaluation results (difference between performance of our method and baseline). Green bars indicate positive values (our method outperforms baseline) and red bars indicate negative values.

4.5 Evaluation

We used two evaluation measures to compare inferred with expert classifications on the test data. The *Adjusted Rand Index* (ARI, Hubert 1985) assesses how much the equivalence relations induced by two partitions coincide. It assumes real values ≤ 1 , where 1 means “perfect agreement” and 0 means “degree of agreement expected by chance”. Negative values may result when from an agreement smaller than expected by chance.

B-Cubed scores (Bagga and Baldwin, 1998) measure precision and recall of a partition analysis compared against a gold standard by computing an individual accuracy score for the cluster decisions on each item in the data and then averaging the results. Hauer and Kondrak (2011) were the first to introduce this measure to test the accuracy of multilingual cognate detection algorithms. In contrast to pair scores such as ARI, B-Cubed scores have the advantage of being independent of the evaluation data itself. While pair-scores tend vary greatly depending on dataset size and cognate density, B-Cubed scores do not show this effect. They are reported as precision and recall. A low B-Cubed precision almost directly translates to the classical notion of a high amount of false positive cognate judgments made by an algorithm, while low B-Cubed recall points to a large amount of cognate sets which were missed by an algorithm.

We took the original LexStat algorithm as a baseline with which we compare our results.

LexStat provides a good baseline, since it was shown to outperform alternative approaches like the above-mentioned CCM approach (Turchin et al., 2010), or clustering based on alternative string similarity measures, like the normalized edit distance, or the normalized scores of the above-mentioned SCA algorithm (List, 2014b). The LexStat implementation in LingPy offers different methods for cognate clustering. Since we employed Infomap for our SVM approach, and since Infomap clustering was shown to work well with LexStat similarities (List et al., 2017), we also used Infomap as the cluster algorithm for the LexStat approach. Since Infomap requires a threshold, we trained the threshold on our training data, excluding short wordlists. Optimal results on the training data was obtained with $\theta^* = 0.57$.

5 Results and Outlook

The evaluation results are given in Table 3, and the differences to the baseline are visualized in Figure 5. On average, the SVM-based classification shows a superior performance when compared to the baseline (an improvement of 0.7% ARI and 0.5% B-cubed F-score). This is mostly due to a substantial improvement for the Austronesian data (4.3% ARI/2.1% B-cubed F-score). Our method slightly outperforms the baseline for Indo-European but is minimally inferior when applied to the Central Asian data. While this might seem a minor improvement only, it is worth exploring on

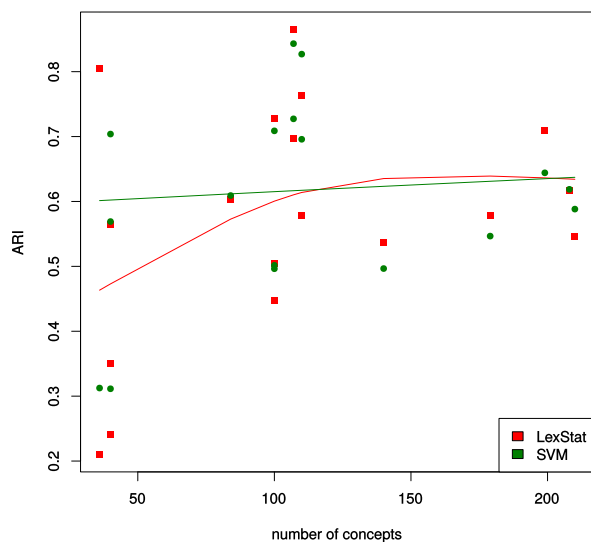


Figure 6: Performance of our method and the benchmark, depending on length of wordlists. Each dot represents one dataset/method pair. The x -axis shows the number of concepts covered in this dataset and the y -axis the Adjusted Rand Index. Solid lines represent smoothed interpolations using Generalized Additive Models.

what type of data our method makes progress.

The plot in Figure 6 shows the dependency of performance (ARI) on the number concepts per data base for the training data. While this result has to be taken with a grain of salt as it involves the data used for model fitting, the pattern is both plausible and striking. It shows that our method clearly outperforms LexStat if the number of concepts is smaller than 100. This finding is unsurprising since LexStat depends on regular sound correspondences. If those cannot be reliably inferred due to data sparseness, its performance drops. Our method is more robust here as it makes use of the PMI string similarity which does not rely on language-specific information. This may also explain the performance on the Austronesian data: although it covers 210 concepts across 100 languages, the languages contain many gaps, and many languages have only 100 words if not even less.

In order to get a clearer impression on where our algorithm failed, we compared false positives and negatives in the Indo-European data (Dunn, 2012), which has been investigated in deep detail during the last 200 years. While a quantitative comparison of part of speech and word length did not reveal any strong correlations with the

accuracy of our approach, a qualitative analysis showed that false positives produced by our approach are usually due to *language-specific factors*. Among the factors triggering false negatives, there are specific *morphological processes* involving complex paradigms, such as Proto-Indo-European **séh₂wel-* ‘sun’, which shows many suffixes in its descendant forms, and specific instances of sound change, involving words that were drastically changed (cf. English *four* vs. French *quatre*). False positives are not only due to chance similarities (compare English *much* with Spanish *mucho*), but also due to words which share morphological elements but are marked as non-cognate in our gold standard (cf. Dutch *man* vs. German *Ehemann* ‘husband’), and errors in the gold standard (cf. Upper Sorbian *powjaz* vs. Lower Sorbian *powrjuz* ‘rope’, wrongly marked as non-cognate in the gold standard).

The classical methods for the identification of cognate words in genetically related languages are based on the general idea that relatedness can be rigorously proven. This requires that the languages under investigation have retained enough similarity to identify regular sound correspondences. The further we go back in time, however, the less similarities we find. The fact that an algorithm like LexStat, which closely mimics the classical comparative method in historical linguistics, needs at least 100 (if not more) concepts in order to yield a satisfying performance reflects this problem of data sparseness in historical linguistics. One could argue that a serious analysis in historical linguistics should never be carried out if data are too sparse. As an alternative to this agnostic attitude, however, one could also try to work on methods that go beyond the classical framework, adding a probabilistic component, where data are too sparse to yield undisputable proof. In this paper, we have tried to make a first step into this direction by testing the power of machine learning approaches with state-of-the-art measures for string similarity in quantitative historical linguistics. The fact that our approach outperforms existing automatic approaches shows that this direction could prove fruitful in future research.

Acknowledgments

This research was supported by the ERC Advanced Grant 324246 EVOLAEMP (GJ, PS), the DFG-KFG 2237 *Words, Bones, Genes, Tools* (GJ),

and the DFG research fellowship grant 261553824 *Vertical and lateral aspects of Chinese dialect history* (JML). We also thank all scholars who contributed to this study by sharing their data.

References

- Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. 2009. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314.
- Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the ACL*, pages 79–85.
- Shane Bergsma and Grzegorz Kondrak. 2007. Multilingual cognate identification using integer linear programming. In *Proceedings of the RANLP Workshop*, pages 656–663.
- Juliana S. Bernardes, Fabio R. J. Vieira, Lygia M. M. Costa, and Gerson Zaverucha. 2015. Evaluation and improvements of clustering algorithms for detecting remote homologous protein families. *BMC Bioinformatics*, 16(1):1–14.
- Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *PNAS*, 110(11):4224–4229.
- Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson. 2012. Mapping the origins and expansion of the Indo-European language family. *Science*, 337(6097):957–960, Aug.
- Claire Bowern and Quentin D. Atkinson. 2012. Computational phylogenetics of the internal structure of pama-nguyan. *Language*, 88:817–845.
- Cecil H. Brown, Eric W. Holman, Søren Wichmann, and Viveka Velupillai. 2008. Automated classification of the world’s languages. *Language Typology and Universals*, 61(4):285–308.
- Cecil H. Brown, Eric W. Holman, and Søren Wichmann. 2013. Sound correspondences in the world’s languages. *Language*, 89(1):4–29.
- Běijīng Dáxué. 1964. *Hányǔ fāngyán cíhuì* [Chinese dialect vocabularies]. Wénzì Gǎigé.
- Michael Cysouw, Søren Wichmann, and David Kamholz. 2006. A critique of the separation base method for genealogical subgrouping. *Journal of Quantitative Linguistics*, 13(2-3):225–264.
- Johannes Dellert and Armin Buch. 2016. Using computational criteria to extract large swadesh lists for lexicostatistics. In Christian Bentz, Gerhard Jger, and Igor Yanovich, editors, *Proceedings of the Leiden Workshop on Capturing Phylogenetic Algorithms for Linguistics*, Tübingen.
- Aron B. Dolgopolsky. 1964. Gipoteza drevnejego rodstva jazykovych semej Severnoj Evrazii s verojatnostej toky zrenija. *Voprosy Jazykoznanija*, 2:53–63.
- Sean S. Downey, Brian Hallmark, Murray P. Cox, Peter Norquest, and Stephen Lansing. 2008. Computational feature-sensitive reconstruction of language relationships. *Journal of Quantitative Linguistics*, 15(4):340–369.
- Michael Dunn. 2012. Indo-European lexical cognacy database (IELex). URL: <http://ielex.mpi.nl/>.
- Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchinson. 2002. *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, 7 edition.
- Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. 2005. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, 6:1889–1918, December.
- Hans Geisler. 1992. *Akzent und Lautwandel in der Romania*. Narr, Tübingen.
- Simon J. Greenhill, Robert Blust, and Russell D. Gray. 2008. The Austronesian Basic Vocabulary Database. *Evolutionary Bioinformatics*, 4:271–283.
- Shirō Hattori. 1973. Japanese dialects. In Henry M. Hoenigswald and Robert H. Langacre, editors, *Diachronic, areal and typological linguistics*, pages 368–400. Mouton, The Hague and Paris.
- Bradley Hauer and Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilingual lists. In *Proceedings of the 5th International Joint NLP conference*, pages 865–873.
- Hóu, Jīngyī, editor. 2004. *Xiàndài Hànyǔ fāngyán yīnkù* [Phonological database of Chinese dialects]. Shànghǎi Jiàoyù, Shànghǎi.
- Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218.
- Gerhard Jäger and Pavel Sofroniev. 2016. Automatic cognate classification with a Support Vector Machine. In Stefanie Dipper, Friedrich Neubarth, and Heike Zinsmeister, editors, *Proceedings of the 13th Conference on Natural Language Processing*, volume 16 of *Bochumer Linguistische Arbeitsberichte*, pages 128–134. Ruhr Universität Bochum.
- Gerhard Jäger. 2013. Phylogenetic inference from word lists using weighted alignment with empirical determined weights. *Language Dynamics and Change*, 3(2):245–291.

- Gerhard Jäger. 2015. Support for linguistic macrofamilies from weighted alignment. *PNAS*, 112(41):12752–12757.
- Brett Kessler. 2001. *The significance of word lists*. CSLI Publications, Stanford.
- Johann-Mattis List and Robert Forkel. 2016. *LingPy 2.5*. Max Planck Institute for the Science of Human History, Jena. URL: <http://lingpy.org>.
- Johann-Mattis List, Michael Cysouw, and Robert Forkel. 2016a. *Concepticon*. Max Planck Institute for the Science of Human History, Jena. URL: <http://concepticon.clld.org>.
- Johann-Mattis List, Philippe Lopez, and Eric Baptiste. 2016b. Using sequence similarity networks to identify partial cognates in multilingual wordlists. In *Proceedings of the ACL 2016 Short Papers*, pages 599–605.
- Johann-Mattis List, Simon Greenhill, and Russell Gray. 2017. The potential of automatic cognate detection for historical linguistics. *PLOS ONE*. DOI: 10.1371/journal.pone.0170046
- Johann-Mattis List. 2014a. Investigating the impact of sample size on cognate detection. *Journal of Language Relationship*, 11:91–101.
- Johann-Mattis List. 2014b. *Sequence comparison in historical linguistics*. Düsseldorf University Press, Düsseldorf.
- Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press, Cambridge, Mass.
- Kenneth A. McElhanon. 1967. Preliminary observations on Huon Peninsula languages. *Oceanic Linguistics*, 6(1):1–45.
- Philippe Menecier, John Nerbonne, Evelyne Heyer, and Franz Manni. 2016. A Central Asian language survey: Collecting data, measuring relatedness and detecting loans. *Language Dynamics and Change*, 6(1).
- Alexander Militarev. 2000. *Towards the chronology of Afrasian (Afroasiatic) and its daughter families*. McDonald Institute for Archaeological Research, Cambridge.
- Saul B. Needleman and Christan D. Wunsch. 1970. A gene method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, July.
- Mark Pagel, Quentin D. Atkinson, and Andrew Meade. 2007. Frequency of word-use predicts rates of lexical evolution throughout Indo-European history. *Nature*, 449(7163):717–720.
- Ilija Peiros. 1998. Comparative linguistics in Southeast Asia. *Pacific Linguistics*, 142.
- John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, pages 61–74. MIT Press.
- Martin Rosvall and Carl T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *PNAS*, 105(4):1118–1123.
- Joy Sanders and Arden G. Sanders. 1980. Dialect survey of the Kamasau language. *Pacific Linguistics. Series A. Occasional Papers*, 56:137.
- Robert R. Sokal and Charles D. Michener. 1958. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438.
- George S. Starostin. 2013. Annotated Swadesh wordlists for the Tujia group. In George S. Starostin, editor, *The Global Lexicostatistical Database*. RGGU, Moscow. URL: <http://starling.rinet.ru>.
- Lydia Steiner, Peter F. Stadler, and Michael Cysouw. 2011. A pipeline for computational historical linguistics. *Language Dynamics and Change*, 1(1):89–127.
- Peter Turchin, Ilya Peiros, and Murray Gell-Mann. 2010. Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, 3:117–126.
- Stijn M. van Dongen. 2000. *Graph clustering by flow simulation*. PhD Thesis, University of Utrecht.
- Feng Wang. 2006. *Comparison of languages in contact. The distillation method and the case of Bai*. Institute of Linguistics Academia Sinica, Taipei.
- Søren Wichmann and Eric W. Holman. 2013. Languages with longer words have more lexical change. In *Approaches to measuring linguistic differences*, pages 249–281. Mouton de Gruyter, Berlin.
- Søren Wichmann, André Müller, Annkathrin Wett, Viveka Velupillai, Julia Bischoffberger, Cecil H. Brown, Eric W. Holman, Sebastian Sauppe, Zarina Molochieva, Pamela Brown, Harald Hammarström, Oleg Belyaev, Johann-Mattis List, Dik Bakker, Dmitry Egorov, Matthias Urban, Robert Mailhammer, Agustina Carrizo, Matthew S. Dryer, Evgenia Korovina, David Beck, Helen Geyer, Pattie Epps, Anthony Grant, and Pilar Valenzuela. 2013. The ASJP Database. Version 16, URL: <http://asjp.clld.org>.
- Mikhail Zhivlov. 2011. Annotated Swadesh wordlists for the Ob-Ugrian group. In George S. Starostin, editor, *The Global Lexicostatistical Database*. RGGU, Moscow. URL: <http://starling.rinet.ru>.
- George K. Zipf. 1935. *The Psycho-Biology of Language*. MIT Press, Cambridge, Massachusetts.

Supplementary Material

The supplementary material can be downloaded from <https://zenodo.org/badge/latestdoi/77850709>. and gives all datasets used for this study along with the results, the source code needed for the replication of the study, and instructions on how to apply the software. If you find errors in the code or want to suggest improvements, please turn to our GitHub repository at <https://github.com/evolaemp/svmcc>. In order to browse through the data and results interactively, have a look at the project website accompanying this publication at <http://www.evolaemp.uni-tuebingen.de/svmcc/>.

A Multi-task Approach to Predict Likability of Books

Suraj Maharjan

Dept. of Computer Science
University of Houston
Houston, TX, 77004
smaharjan2@uh.edu

John Arevalo and **Fabio A. González**

Computing Systems and
Industrial Engineering Dept.
Universidad Nacional de Colombia
Bogotá, Colombia
{jearevaloo, fagonzalezo}@unal.edu.co

Manuel Montes-y-Gómez

Instituto Nacional de Astrofísica
Óptica y Electrónica
Puebla, Mexico
mmontesg@ccc.inoep.mx

Thamar Solorio

Dept. of Computer Science
University of Houston
Houston, TX, 77004
solorio@cs.uh.edu

Abstract

We investigate the value of feature engineering and neural network models for predicting successful writing. Similar to previous work, we treat this as a binary classification task and explore new strategies to automatically learn representations from book contents. We evaluate our feature set on two different corpora created from Project Gutenberg books. The first presents a novel approach for generating the gold standard labels for the task and the other is based on prior research. Using a combination of hand-crafted and recurrent neural network learned representations in a dual learning setting, we obtain the best performance of 73.50% weighted F1-score.

1 Introduction

Every year millions of new books are published, but only a few of them turn into commercial successes, and even fewer achieve critical praise in the form of prestigious awards or meaningful sales. Editors have the difficult task of making the go/no-go decision for all manuscripts they receive, and the revenue for their publishing house depends on the accuracy of that judgment. The website www.litrejections.com documents some of the biggest mistakes in the history of the publishing industry, including Agatha Christie, J.K. Rowling, and Dr. Seuss, all of whom received many rejection letters before landing their first publishing deal.

Many factors contribute to the eventual success of a given book. Internal factors such as plot, story line, and character development all have a role in the likability of a book. External factors such as author reputation and marketing strategy are arguably equally relevant. Some factors might even be out of the control of an author or publishing house, such as the current trends, the competition from books released simultaneously, and the historical and contextual factors inherent to society.

Previous work by Ganjigunte Ashok et al. (2013) demonstrated relevant results using stylistic features to predict the success of books. Their definition of success was a function of the number of downloads from Project Gutenberg. However downloading a book is not by itself an indicator of a highly liked or a commercially successful book. We instead propose to use the rating from reviewers collected from Goodreads as a measure of success. We also propose features and deep learning techniques that have not been used before on this problem, and validate their usefulness in two different tasks: success prediction and genre classification. Our key contributions are the following:

- We provide a new benchmark dataset for predicting successful books in a more realistic class distribution. This data set is available to the community from this link¹.
- We show that sentiment analysis using SenticNet sentics is an accurate way to model emotion in books.

¹The data can be downloaded from <http://ritual.uh.edu/resources/page>.

- We provide the first results on using recurrent neural networks (RNN) to discover book content representations that are useful for classification tasks such as success prediction and genre detection.
- We show that the multitask approach, simultaneously evaluating success and genre prediction, benefits from its constituent tasks to obtain better performance than the single success prediction task approach.

2 Previous work

Predicting the success of books is a difficult task, even for an experienced editor. Researchers have studied related tasks, for example predicting the quality of text from lexical features, syntactic features and different measures of density. Pitler and Nenkova (2008) found a strong correlation between user-perceived text quality and the likelihood measures of the vocabulary as computed by a language model, as well as the likelihood measures of discourse relations, as determined by a language model trained on discourse relations. Louis and Nenkova (2013) proposed a combination of genre-specific and readability features with topic-interest metrics for the prediction of great writing in science articles. While some of the features in this prior work were relevant to our task, our goal is different and more aligned to Ganjigunte Ashok et al. (2013), since we aim to model success in books of different genres.

Ganjigunte Ashok et al. (2013) investigated the correlation between writing style and number of downloads. The authors analyzed lexical features, production rules, constituents, and sentiment features of books downloaded from Project Gutenberg². They obtained an average accuracy of 70.38% using only unigram features with Support Vector Machines (SVM) as the classifier.

Deep learning representations have seen their share of successes in Natural Language Processing (NLP) tasks (Bahdanau et al., 2014; Zheng et al., 2013; Gao et al., 2014; Glorot et al., 2011; Samih et al., 2016). In particular, RNN models have been successfully applied in several scenarios where temporal dependencies provide relevant information (Ian Goodfellow and Courville, 2016; LeCun et al., 2015). Kiros et al. (2015) used RNN models to learn language

models from books using an unsupervised approach. Also, word embedding (Mikolov et al., 2013) and Paragraph Vector (Le and Mikolov, 2014) have been shown to achieve state-of-the-art performance in several text classification and sentiment classification tasks. These techniques are able to learn distributed vector representations that capture semantic and syntactic relationships between words. Collobert and Weston (2008) trained jointly a single Convolutional Neural Network (CNN) architecture on different NLP tasks and showed that multitask learning increases the generalization of the shared tasks. Other researchers (Ian Goodfellow and Courville, 2016; Søggaard and Goldberg, 2016; Attia et al., 2016) have also reached to similar conclusions.

3 Dataset

We experimented with two book collections: one prepared by Ganjigunte Ashok et al. (2013)³ and the other constructed by us to evaluate a new definition of success. We refer to the first dataset as EMNLP13 and the second dataset as Goodreads.

The EMNLP13 collection contained Project Gutenberg books from eight different genres. The authors created a balanced dataset containing 100 books per genre, resulting in a total of 800 books. We manually reviewed the dataset and found missing or irrelevant content in 58 books: a total of 53 books contained Project Gutenberg license information repeated verbatim, and five books contained only the audio recording certificate in place of the actual book content. We removed the license-related text, since lexical features might be erroneously biased, and replaced the five files with the actual content of the books. Except for these corrections, the data we used is the same as that presented in Ganjigunte Ashok et al. (2013).

We also identified some odd adjudications. For example, ‘The Prince And The Pauper’ is a popular book by Mark Twain that was adapted into various films and stage plays. Also, ‘The Adventures of Captain Horn’ was the third best selling book of 1895 (Hackett, 1967). Both these books are labeled as unsuccessful due to their low download counts. We suspect as well that some of the counts are inflated by college students doing English or Literature assignments that may not be directly related to the potential commercial success

² <https://www.gutenberg.org/>

³The data can be downloaded from <http://www3.cs.stonybrook.edu/~songfeng/success/>

Genre	Unsuccessful	Successful	Total
Detective Mystery	60	46	106
Drama	29	70	99
Fiction	30	81	111
Historical Fiction	16	65	81
Love Stories	20	60	80
Poetry	23	158	181
Science Fiction	48	39	87
Short Stories	123	135	258
Total	349	654	1,003

Table 1: Goodreads Data Distribution

		EMNLP13 Success definition	
		Unsuccessful	Successful
Goodreads Success definition	Unsuccessful	73	32
	Successful	110	184

Table 2: Confusion matrix between two different definitions of success.

of a book.

To address these concerns, we propose a new approach to creating gold labels for successful books based on public reviews rather than download counts. We collected a new set of Project Gutenberg books for this benchmarking. We mapped the books to their review pages on Goodreads⁴, a website where book lovers can search, review, and rate books. We consider only those books that have been rated by at least 10 people. We use the average star rating and total number of reviews for labeling each book. We then set an average rating of 3.5 as the threshold for success, such that books with average rating < 3.5 are classified as *Unsuccessful*. Table 1 shows the data distribution of our books. To our knowledge, we have one of the largest collection of books, as researchers generally work with a low number of books (Coll Ardanuy and Sporleder, 2014; Goyal et al., 2010; van Cranenburgh and Koolen, 2015).

Success Definitions Comparison: After compiling and labeling both the datasets, we drew a comparison between the two definitions of success. To do this, we downloaded the Project Gutenberg download counts for the books in Goodreads dataset and labeled them using the Ganjigunte Ashok et al. (2013) definition of success. Since they only considered books in the extremes of download counts, we could only label 399 books in the Goodreads dataset using their definition. We found that 142 books had different labels according to the two definitions. 19.7% of these mismatched books were labeled as unsuccessful

despite having ratings ≥ 3.5 and being reviewed by more than 100 reviewers. Table 2 details the discrepancies between the two definitions.

4 Methodology

We investigated a wide range of textual features in an attempt to capture the topic, sentiment, writing style, and readability for each book. This set included both new and previously used features. We also explored techniques for automatically learning representations from text using neural networks, which have been shown to be successful in various text classification tasks (Kiros et al., 2015; LeCun et al., 2015). These techniques include word embeddings, document embeddings, and recurrent neural networks.

4.1 Hand-crafted text features

Lexical: We used skip-grams, char n -grams, and typed char n -grams (Sapkota et al., 2015) with term frequency-inverse document frequency (TF-IDF) as the weighting scheme. Sapkota et al. (2015) showed that classical character n -grams lose some information in merging instances of n -grams like *the* which could be a prefix (*thesis*), a suffix (*breathe*), or a standalone word (*the*). They separated character n -grams into ten categories representing grammatical classes, like affixes, and stylistic classes, like beg-punct and mid-punct which reflect the position of punctuation marks in the n -gram. The purpose of these features is to correlate success with an author’s word choice.

Constituents: We computed the normalized counts of ‘*SBAR*’, ‘*SQ*’, ‘*SBARQ*’, ‘*SINV*’, and ‘*S*’ syntactic tag sets from the parse tree of each sentence in each book, following the method of Ganjigunte Ashok et al. (2013) to determine the syntactic style of the authors.

Sentiment: We computed sentence neutrality, positive and negative, using SentiWordNet (Baccianella et al., 2010) along with the counts of nouns, verbs, adverbs, and adjectives. We averaged these scores for every 50 consecutive sentences in order to evaluate change in sentiment throughout the course of each book, because we anticipate emotions, like suspense, anger, and happiness to contribute to the success of the book.

SenticNet Concepts: We extracted sentiment concepts from the books using the Sentic Concept

⁴<https://www.goodreads.com/>

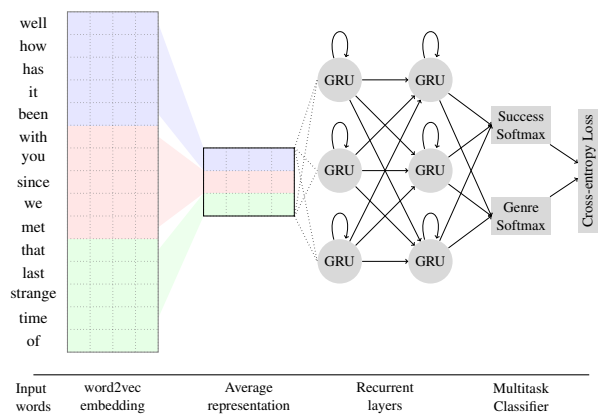


Figure 1: Multitask method. Words are represented in the Word2Vec space. Such representations are averaged per window. Sequences are feed to GRU network. Finally, the features are feed to two softmax components to predict genre and success simultaneously.

Parser⁵. The parser chunks a sentence into noun and verb clauses, and extracts concepts from them using Part Of Speech (POS) bigram rules. We modeled these as binary bag-of-concepts (BoC) features. We also extracted average polarity, sensitivity, attention, pleasantness, and aptitude scores for the concepts defined in the SenticNet-3.0 knowledgebase, which contains semantics and sentsics associated with 30,000 common-sense concepts (Cambria and Hussain, 2015).

Writing density: We computed the number of words, characters, uppercase words, exclamations, question marks, as well as the average word length, sentence length, words per sentence, and lexical diversity of each book, with the expectation that successful and unsuccessful writings will have dissimilar distributions of these density metrics.

Readability: We computed multiple readability measures including Gunning Fog Index (Gunning, 1952), Flesch Reading Ease (Flesch, 1948), Flesch Kincaid Grade Level (Kincaid et al., 1975), RIX, LIX (Anderson, 1983), ARI (Senter and Smith, 1967), and Smog Index (Mc Laughlin, 1969) and used their mean normalized values for training. Intuitively, the use of simple language will resonate with a larger audience and contribute to book success.

4.2 Neural network learned representations

Representation learning techniques are able to learn a set of features automatically from the raw data. Our hypothesis is that the learned representation can capture the complex factors that influence the success of a book.

Word embeddings with Book2Vec: In contrast with Word2Vec, which learns a representation for individual words, Doc2Vec learns a representation for text fragments or even for full documents. We trained the Doc2Vec module of the Gensim (Řehůřek and Sojka, 2010) Python library, on all the books in the Goodreads dataset to obtain a 500 dimensional dense vector representation for each book. Using Doc2Vec, we first trained a distributional memory (DM) model with two approaches: concatenation of context vectors (DMC) and sum of context word vectors (DMM). Then we trained a distributional bag of words (DBoW) model and combined it with the DMC and the DMM for a total of five different models. We set the number of iterations to 50 epochs and shuffled the training data in each pass. We called these book vectors *Book2Vec*. Furthermore, we created two 300 dimensional vector representations for each book by averaging the vectors of each word in the book using pre-trained Word2Vec vectors from the Google News dataset⁶ and our own Word2Vec trained with $\sim 350M$ words from 5,000 random books crawled from Project Gutenberg.

Multitask RNN method: When dealing with variable length data such as time series or plain text, traditional approaches like feed-forward neural networks are not easily adapted since they expect fixed-size input to model sequential data. One limitation of RNNs is that it has problems dealing with long sequences (Pascanu et al., 2013). We propose a strategy to represent large documents, such as books, with an aggregated representation. Figure 1 depicts the proposed multitask method. The overall strategy uses a RNN to learn a model of sequences of sentences. Each sentence is represented by the average of the Word2Vec representation of its constituent words. The RNN is composed of 2 hidden layers with 32 hidden gated recurrent units (GRU) (Cho et al., 2014) each, and the output is a softmax layer. We train the RNN

⁵<https://github.com/pbhuss/Sentimental/blob/master/parser/SenticParser.py>

⁶The pre-trained Word2Vec was downloaded from <https://code.google.com/p/word2vec/>

in a supervised fashion using the success categorization and the book genre as labels. The RNN serves a feature extractor and the last hidden states for each sequence acts as its representation. At training time, all sentences from one book are extracted and divided in chunks of 128 sentences. The book’s success/genre labels are assigned to each sequence. A sentence is then represented as the average of its constituent word vectors. To make the book label assignment at testing time, we average the predictions of all sequences extracted from each book. Using 128 sentences has three-fold a motivation: (1) mitigate vanishing gradient problem (Pascanu et al., 2013), (2) obtain more examples from one book, and (c) be a power of 2 to efficiently use the GPU.

An interesting property of neural networks is that the same learning approach, i.e stochastic gradient descent, still holds for more complex architectures as long as the objective cost function is differentiable. We take advantage of this property to build a unified neural network that addresses both genre and success prediction using a single model. These kinds of multitask architectures are also useful as regularizers (Ian Goodfellow and Courville, 2016). In particular, our cost function $J(X, Y)$ is defined as follows:

$$\begin{aligned}
 h_i &= rnn(x_i) \\
 \hat{y}_i^{succ} &= \frac{e^{z_i^{succ}}}{\sum_k e^{z_k^{succ}}} \\
 \hat{y}_i^{gen} &= \frac{e^{z_i^{gen}}}{\sum_l e^{z_l^{gen}}} \\
 J(X, Y) &= - \sum_i (y_i^{succ} \ln \hat{y}_i^{succ} + y_i^{gen} \ln \hat{y}_i^{gen})
 \end{aligned}$$

where x_i represents the i -th sample and y^{succ} and y^{gen} are success and genre labels respectively. The $rnn(\cdot)$ function represents the forward propagation over the recurrent neural network and h represents the last hidden state. \hat{y}^{succ} and \hat{y}^{gen} represent predictions for the two labels. Notice that both of them are computed using the same unified representation h . z^{succ} and z^{gen} represent two different linear transformations over h that map to the number of classes.

5 Experiments and Results

5.1 Experiments on Goodreads dataset

We merged books from different genres, and then randomly divided the data into a 70:30 train-

ing/test ratio, while maintaining the distribution of *Successful* and *Unsuccessful* classes per genre. As a preprocessing step we converted all words to lowercase and removed infrequent tokens having document frequency ≤ 2 . For our tagging and parsing needs, we used the Stanford parser (Socher et al., 2013). We then trained a LibLinear Support Vector Machine (SVM)⁷ classifier with L2 regularization using the hand-crafted features described in Section 4. We tuned the C parameter in the training set with 3-fold grid search cross-validation over different values of $1e\{-4, \dots, 4\}$.

With the features used by Ganjigunte Ashok et al. (2013), we obtained the highest weighted F1-score of 0.659 with word bigram features. We set this value as our baseline. In order to study the effect of the multitask approach, we devised analogous experiments to our proposed multitask RNN method and predicted both genre and success together for the features described in Section 4. Hence we have two settings for the classification experiments, Single task (ST) and Multitask (MT).

Since we had average rating information, we also modeled the problem as a regression problem and predicted the average rating using only the content of the books. Our work differs from other researchers in this aspect, as most of them (Lei et al., 2016; Li et al., 2011; Mudambi et al., 2014) use review content instead of the actual book content to predict the average rating. We used the Elastic Net regression algorithm with *l1_ratio* tuned over range $\{0.01, 0.05, 0.25, 0.5, 0.75, 0.95, 0.99\}$ with 3-fold grid search cross-validation of the training data.

Parameter tuning for RNN: We trained 25 models with random hyper-parameter initialization for learning rate, weights initialization ranges and regularization parameters. We chose the best validation performance model. This is preferable over grid search when training deep models (Bergstra and Bengio, 2012). We used the ADAM algorithm (Kingma and Ba, 2014) to update the gradients. Since these models are prone to overfitting because of the high number of parameters, we applied clip gradient, max-norm weights, early stopping and dropout regularization strategies.

⁷We use LibLinear SVM wrapper from <http://scikit-learn.org/stable/>

Features	ST (F1)	MT (F1)	MSE
Word Bigram	0.659	0.685	0.152
2 Skip 2 gram	0.645	0.688	0.156
2 Skip 3 gram	0.506	0.680	0.156
Char 3 gram	0.669	0.700	0.155
Char 4 gram	0.676	0.689	0.155
Char 5 gram	0.683	0.699	0.154
Typed beg_punct 3 gram	0.621	0.672	0.151
Typed mid_punct 3 gram	0.598	0.641	0.151
Typed end_punct 3 gram	0.626	0.677	0.151
Typed mid_word 3 gram	0.653	0.687	0.156
Typed whole_word 3 gram	0.658	0.666	0.154
Typed multi_word 3 gram	0.607	0.657	0.154
Typed prefix 3 gram	0.624	0.624	0.154
Typed space_prefix 3 gram	0.589	0.646	0.155
Typed suffix 3 gram	0.624	0.637	0.154
Typed space_suffix 3 gram	0.626	0.664	0.154
Clausal	0.506	0.558	0.156
Writing Density (WR)	0.605	0.640	0.156
Readability (R)	0.506	0.634	0.144
SentiWordNet Sentiments(SWN)	0.582	0.610	0.156
Sentic Concepts and Scores (SCS)	0.657	0.670	0.155
GoogleNews Word2Vec	0.669	0.692	0.156
Gutenberg Word2Vec	0.672	0.673	0.140
Book2Vec (DBoW)	0.643	0.654	0.130
Book2Vec (DMM)	0.686	0.731	0.142
Book2Vec (DMC)	0.640	0.674	0.131
Book2Vec (DBoW+DMC)	0.647	0.677	0.131
Book2Vec (DBoW+DMM)	0.695	0.729	0.142
RNN	0.529	0.686	0.125

Table 3: Results for classification (ST = Single task setting, MT = Multi-task setting) and regression tasks on Goodreads dataset. MSE = Mean Square Error, F1 score is weighted F1 scores across *Successful* and *Unsuccessful* classes.

5.2 Results on Goodreads dataset

Table 3 shows the results with our new proposed feature sets for the classification and regression tasks. In the ST setting, except for the character n -gram features, all proposed hand-crafted features individually had a weighted F1-score less than the word bigram baseline. On the other hand, the neural network methods obtained better results than the baseline. We obtained the highest weighted F1-score of 0.695 and 0.731 with the *Book2Vec* method in the ST and MT settings, respectively. The results show that the MT approach is better than the ST approach. The genre prediction task must have acted as a regularizer for the success prediction task. Also, we found that modeling the entire book as a vector, rather than modeling it as the average of word vectors, gave better performance. Although the ST *Book2Vec* performs better than the MT RNN method, the difference is very small. We performed McNemar’s test on these methods and found that the results were not statistically significant, with $p=0.5$. The MT RNN method had the lowest mean square error (MSE) for the regression task, at 0.125.

The character n gram proved to be one of the most important hand-crafted features, whereas clausal feature was the least important one. In-

Features	ST (F1)	MT (F1)	MSE
Unigram+Bigram	0.660	0.691	0.15
Unigram+Bigram+Trigram	0.660	0.700	0.149
Char 3,4,5 gram	0.682	0.689	0.153
All Typed ngram	0.663	0.691	0.144
SCS+WR+Typed mid word	0.720	0.710	0.155
SCS+Book2Vec	0.695	0.731	0.139
R+Book2Vec	0.695	0.729	0.139
WR+Book2Vec	0.693	0.726	0.139
Word Ngram+ RNN	0.691	0.688	0.125
Skip gram + RNN	0.689	0.683	0.125
Typed char ngram+ RNN	0.689	0.702	0.125
Char 3 gram + RNN	0.689	0.688	0.125
Clausal+ RNN	0.689	0.688	0.125
SCS + RNN	0.691	0.688	0.125
WR+Book2Vec+ RNN	0.701	0.735	0.129
SCS+WR+RNN	0.675	0.696	0.123
All hand-crafted	0.670	0.689	0.148
All hand-crafted+neural	0.667	0.712	0.129

Table 4: Feature Combination Results for Goodreads dataset. (ST = Single Task, MT =Multi-task, SCS = Sentic concept+average scores of sensitivity, attention, pleasantness, aptitude, polarity, WR = Writing Density, R = Readability)

dividually, writing density and readability features seemed to be weak features. We assumed that the sentiment changes in books would be an important characteristic for the task. However, the results in Table 3 show an unimpressive F1-score of 0.610 for sentiment features. On the other hand, the bag of sentic concepts model with average scores for sensitivity, attention, pleasantness, aptitude, and polarity gave a more impressive F1-score of 0.670, much higher than the baseline. This result points to the relevance of performing a more nuanced sentiment analysis beyond lexical statistics for this task.

Our next set of experiments included the combinations of hand-crafted and neural network representations. Some of the best combination results are shown in Table 4. Out of the different possible feature combinations, we obtained the highest weighted F1 score of 0.735 by combining hand-crafted and learned representations in the MT setting. We observed that combining low performing hand-crafted features like readability, syntactic clauses, and skip grams with neural representation boosted their performance. Likewise for the regression task, the MT RNN representation proved to be a better choice, as its combination with other features generally lowered the MSE. The best combinations for the regression task lowered the MSE to 0.123. Deep learning and hand-crafted methods may capture complementary sources of information, which upon combination boost performance.

5.3 Results on EMNLP13 dataset

We tried to reproduce the results reported in Ganjigunte Ashok et al. (2013) by re-implementing their system. Unlike our setup, they performed experiments on individual genres and reported average accuracy across all genres. We obtained similar results, but not as close as we expected, even after extensive experimentation, and extending the search for parameter optimization. For most of their features we obtained a lower accuracy⁸. The differences may be due to a combination of the curating process we described in Section 3 that corrected content in the books used, as well as the different set of parameter values we explored for tuning the classifier. As pointed out by Fokkens et al. (2013), even seemingly small differences in preprocessing can prevent reproducibility. *Hence, we consider our best accuracy so far (71.25%) to be the state-of-the-art performance on this data set.*

Table 5 shows the results from some of our best feature sets. The features that worked best for the Goodreads data also worked best for the EMNLP13 data. Significantly, with the combination of the sentic concepts and scores, typed *n*grams, and writing density, we obtained an average accuracy of 73.00%, much higher than the baseline score of 71.25% for this dataset.

The RNN performance was very low in comparison with the handcrafted features. We relate this behavior to the small size of this particular training dataset and evaluation setup. Notice that Ganjigunte Ashok et al. (2013) experimented per genre, i.e. trained a single classifier per genre. Thus, in a 5-fold approach we only have 80 samples to train and 20 to test. Additionally, we must take out some samples from the training data for validation. It has been empirically shown that one of the key elements in the success of representation learning strategies is a large amount of data, on the order of tens of thousands of samples at least. Moreover, in the EMNLP13 dataset, it is not possible to take advantage of the multitask approach because there is only one target genre in each experiment.

6 Discriminative features

Table 6 lists some of the features that were highly-weighted by the classifier. For the sentic concepts, salient features included important adjectives, verbs and relations; all objects that might

⁸There was a maximum 4% difference for some features.

Features	Avg Accuracy(%)
Word Bigram	71.25
Char 3 grams	71.00
Typed mid_word 3-gram	70.25
Writing Density (WR)	68.38
Readability	61.38
Sentic concepts & scores(SCS)	72.38
GoogleNews Word2Vec	69.88
Gutenberg Word2Vec	64.25
Book2Vec	72.38
RNN	55.80
Unigram+Bigram+Trigram	72.75
Book2Vec+SCS	64.75
Book2Vec+WR	66.38
SCS+WR+Typed char ngrams	73.00

Table 5: Average accuracy results with new feature and their combinations on EMNLP13 dataset.

Type	Features
ngrams	. " . " , said , young man, very young man, the young man, boys, . i, father, his father, mother, he said, she said, said NE, princess, lord, colonel, captain, doctor, tour, mr, miss
Sentic concepts	conceive, grieve, zealous, emptiness, bitterness, corpse, hypothesis, irony, theory_of.the, wagon,deep,blue, scarred, screaming, grudging, vigil, vein, beautiful.place, rural, marriage, friendship, cats, 911 avg aptitude, polarity, pleasantness, attention scores
Character and typed character ngrams	mr., mrs., john, thou, amor, pen, his, and,the, ing, n's,ed, gg', pt', d'a, t', i-t, .., 'i, " " , " say," s," she

Table 6: Discriminative Features

trigger a crucial event. Similarly, for the character *n*-gram features, honorific titles, stop words, common word endings, and especially *n*-grams with quotation marks were highly weighted. Quotation marks indicate the exchange of dialogues between characters. This suggests that dialogue is an important aspect of novels. Word *n*-gram features also support this suggestion. Features like *s/he said, said Person.Name* were also highly weighted. Moreover, pronouns and titles related to male gender also had high weights. Features like *i was, . i, i am* also had high weights. This might be an indication that books with first person narration tend to be more successful. Another interesting observation was that the number of question marks in a book was also consistently positively correlated with success. This might suggest that readers enjoy books consisting of dialogue or interaction between the characters. We also calculated the maximal information criterion (MIC) and correlation coefficient (CC) for the writing density as well as the readability features against the average rating. Generally, readers prefer books with high writing density (0.19 MIC, 0.25 CC) and somewhat complex writing (0.17 MIC, 0.21 CC).

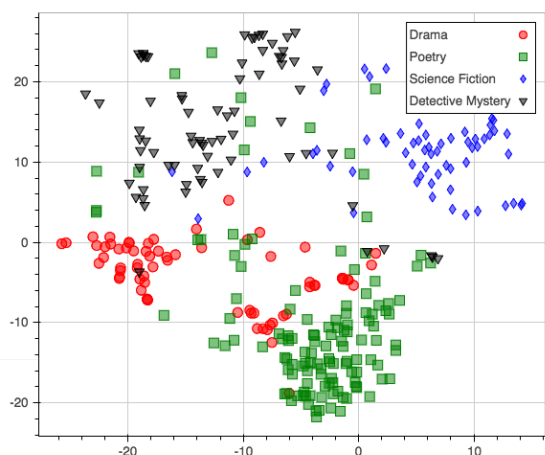


Figure 2: Projection of Book2Vec from four different genres into 2D space for the Goodreads dataset.



Figure 3: Projection of successful and unsuccessful books using representation learned with the RNN model.

7 Analysis of learned representations

In order to investigate deep vectors, we projected them onto 2-dimensional space using t-SNE. Figure 2 suggests that the vectors successfully capture genre-related concepts, as books from the same genre are close to each other in the 2D space. We then performed 8-way genre classification experiment using random stratified division of the data into 70:30 training/test ratio. We obtained an accuracy of 62.50% and F1 score of 69.30% for the EMNLP13 and Goodreads datasets, respectively. These scores were well above the random baseline of 12.50% accuracy and 15.23% F1-score for the EMNLP13 and Goodreads datasets, respectively. We further found that Poetry and Drama were the most accurately classified genres, whereas Fiction was the most difficult to classify.

In order to further investigate the representations learned by RNN for successful and unsuccessful books, we plotted the 2D t-SNE projection of the book representations. Figure 3 shows the

projection of vectors for the Short stories genre. The visualization shows that the RNN is able to cluster the book vectors into two separate regions. Furthermore, to investigate what else the RNN might be learning, we plotted some books by the same authors. Figure 3 also shows books from authors Jack London and Alan E. Nourse. The four books by Jack London and the two books by Alan E. Nourse are very close to each other. We thus infer that along with learning peculiarities of successful and unsuccessful classes, the RNN was able to capture features related to the style of authors.

8 How much content is needed for success prediction?

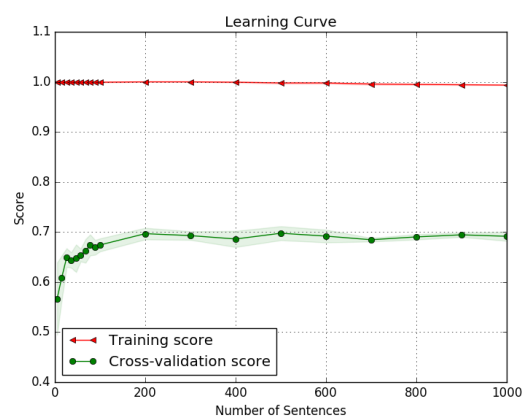


Figure 4: Weighted F1 score for training and validation data for varying number of sentences with char 3 gram feature.

Humans are good at detecting poor writing after reading just a few pages. We wanted to investigate if it is the same for machines. We devised stratified 3-fold cross-validation exploratory experiments on training data by gradually increasing the content of the books in the training fold. The results are shown in Figure 4. We see that the cross-validation score gradually increases until we reach 200 sentences. After this point, it plateaued out. Hence, we conclude that 200 sentences is the minimum threshold for the classifier.

9 Conclusions

In this paper we propose new features for predicting the success of books. We used two main feature categories: hand-crafted and RNN-learned features. Hand-crafted features included typed character n -grams and sentic concepts. For the

learned features we proposed two different strategies based on neural networks. The first extends Word2Vec-type representations to work in large documents such as books, and the second one uses an RNN to capture sequential patterns in large texts. We evaluated our methods on our Goodreads dataset, whose classes are not based on download counts, but rather are a function of average star ratings and number of reviewers. Our results outperform state-of-the-art methods. We conclude that instead of having either deep-learning or hand-crafted features outperform the other, both methods capture complementary information, which upon combination gives better performance. Also, the multitask setting is preferable to the single task setting, as the multitask approach helps the classifier better generalize during learning by letting constituent tasks act as regularizers. As our next steps, we plan to investigate features that capture plot-related aspects, such as character profiles and interaction through social network analysis, historical setting, and other feature-learning strategies.

Acknowledgments

We would like to thank the National Science Foundation for partially funding this work under awards 1462141 and 1549549. We also thank Simon Tice and the three anonymous reviewers for reviewing the paper and providing helpful comments and suggestions.

References

- Jonathan Anderson. 1983. Lix and rix: Variations on a little-known readability index. *Journal of Reading*, 26(6):490–496.
- Mohammed Attia, Suraj Maharjan, Younes Samih, Laura Kallmeyer, and Thamar Solorio. 2016. Cogalex-v shared task: Ghhh - detecting semantic relations via word embeddings. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 86–91, Osaka, Japan, December. The COLING 2016 Organizing Committee.
- Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.
- James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.
- Erik Cambria and Amir Hussain. 2015. *Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis*, volume 1. Springer.
- Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October. Association for Computational Linguistics.
- Mariona Coll Ardanuy and Caroline Sporleder. 2014. Structure-based clustering of novels. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 31–39, Gothenburg, Sweden, April. Association for Computational Linguistics.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- Rudolph Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–223.
- Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1764, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 699–709, Baltimore, Maryland, June. Association for Computational Linguistics.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

- Amit Goyal, Ellen Riloff, and Hal Daume III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86, Cambridge, MA, October. Association for Computational Linguistics.
- Robert Gunning. 1952. *The technique of clear writing*. McGraw-Hill.
- Alice Payne Hackett. 1967. *Seventy years of best sellers, 1895-1965*. RR Bowker Co.
- Yoshua Bengio Ian Goodfellow and Aaron Courville. 2016. Deep learning. Book in preparation for MIT Press.
- J. Peter Kincaid, Robert P. Fishburne Jr, Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.
- Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.
- Xiaojiang Lei, Xueming Qian, and Guoshuai Zhao. 2016. Rating prediction based on social sentiment from textual reviews. *IEEE Transactions on Multimedia*, 18(9):1910–1921.
- Fangtao Li, Nathan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, and Xiaoyan Zhu. 2011. Incorporating reviewer and product information for review rating prediction. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI’11*, pages 1820–1825. AAAI Press.
- Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics*, 1:341–352.
- G. Harry Mc Laughlin. 1969. Smog grading-a new readability formula. *Journal of reading*, 12(8):639–646.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR), Workshop*.
- Susan M. Mudambi, David Schuff, and Zhewei Zhang. 2014. Why aren’t the stars aligned? an analysis of online review content and star ratings. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 3139–3147. IEEE.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318.
- Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195, Honolulu, Hawaii, October. Association for Computational Linguistics.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. <http://is.muni.cz/publication/884893/en>.
- Younes Samih, Suraj Maharjan, Mohammed Attia, Laura Kallmeyer, and Thamar Solorio. 2016. Multilingual code-switching identification via lstm recurrent neural networks. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 50–59, Austin, Texas, November. Association for Computational Linguistics.
- Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, Denver, Colorado, May–June. Association for Computational Linguistics.
- R.J. Senter and E.A. Smith. 1967. Automated readability index. Technical report, DTIC Document.
- Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August. Association for Computational Linguistics.
- Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235,

Berlin, Germany, August. Association for Computational Linguistics.

Andreas van Cranenburgh and Corina Koolen. 2015. Identifying literary texts with bigrams. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 58–67, Denver, Colorado, USA, June. Association for Computational Linguistics.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA, October. Association for Computational Linguistics.

A Data-Oriented Model of Literary Language

Andreas van Cranenburgh

Institut für Sprache und Information
Heinrich Heine University Düsseldorf
cranenburgh@phil.hhu.de

Rens Bod

Institute for Logic, Language and
Computation, University of Amsterdam
rens.bod@uva.nl

Abstract

We consider the task of predicting how literary a text is, with a gold standard from human ratings. Aside from a standard bigram baseline, we apply rich syntactic tree fragments, mined from the training set, and a series of hand-picked features. Our model is the first to distinguish degrees of highly and less literary novels using a variety of lexical and syntactic features, and explains 76.0 % of the variation in literary ratings.

1 Introduction

What makes a literary novel *literary*? This seems first of all to be a value judgment; but to what extent is this judgment arbitrary, determined by social factors, or predictable as a function of the text? The last explanation is associated with the concept of *literariness*, the hypothesized linguistic and formal properties that distinguish literary language from other language (Baldick, 2008). Although the definition and demarcation of literature is fundamental to the field of literary studies, it has received surprisingly little empirical study. Common wisdom has it that literary distinction is attributed in social communication about novels and that it lies mostly outside of the text itself (Bourdieu, 1996), but an increasing number of studies argue that in addition to social and historical explanations, textual features of various complexity may also contribute to the perception of literature by readers (cf. Harris, 1995; McDonald, 2007). The current paper shows that not only lexical features but also hierarchical syntactic features and other textual characteristics contribute to explaining judgments of literature.

Our main goal in this project is to answer the following question: are there particular textual conventions in literary novels that contribute to readers judging them to be literary? We address this ques-

tion by building a model of literary evaluation to estimate the contribution of textual factors. This task has been considered before with a smaller set of novels (restricted to thrillers and literary novels), using bigrams (van Cranenburgh and Koolen, 2015). We extend this work by testing on a larger, more diverse corpus, and by applying rich syntactic features and several hand-picked features to the task. This task is first of all relevant to literary studies—to reveal to what extent literature is empirically associated with textual characteristics. However, practical applications are also possible; e.g., an automated model could help a literary publisher decide whether the work of a new author fits its audience; or it could be used as part of a recommender system for readers.

Literary language is arguably a subjective notion. A gold standard could be based on the expert opinions of critics and literary prizes, but we can also consider the reader directly, which, in the form of a crowdsourced survey, more easily provides a statistically adequate number of responses. We therefore base our gold standard on a large online survey of readers with ratings of novels.

Literature comprises some of the most rich and sophisticated language, yet stylometry typically does not exploit linguistic information beyond part-of-speech (POS) tags or grammar productions, when syntax is involved at all (cf. e.g., Stamatatos et al., 2009; Ashok et al., 2013). While our results confirm that simple features are highly effective, we also employ full syntactic analyses and argue for their usefulness. We consider tree fragments: arbitrarily-sized connected subgraphs of parse trees (Swanson and Charniak, 2012; Bergsma et al., 2012; van Cranenburgh, 2012). Such features are central to the Data-Oriented Parsing framework (Scha, 1990; Bod, 1992), which postulates that language use derives from arbitrary chunks (e.g., syntactic tree fragments) of previous lan-

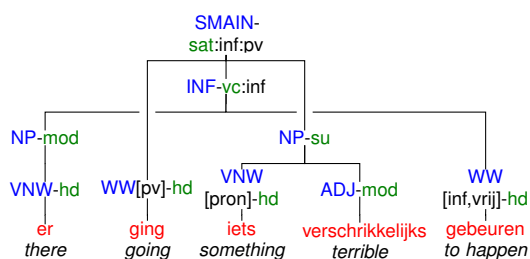


Figure 1: A parse tree fragment from Franzen, *The Corrections*. Original sentence: something terrible was going to happen.

guage experience. In our case, this suggests the following hypothesis.

HYPOTHESIS 1: Literary authors employ a distinctive inventory of lexico-syntactic constructions (e.g., a register) that marks literary language.

Next we provide an analysis of these constructions which supports our second hypothesis.

HYPOTHESIS 2: Literary language invokes a larger set of syntactic constructions when compared to the language of non-literary novels, and therefore more variety is observed in the parse tree fragments whose occurrence frequencies are correlated with literary ratings.

The support provided for these hypotheses suggests that the notion of literature can be explained, to a substantial extent, from textual factors, which contradicts the belief that external, social factors are more dominant than internal, textual factors.

2 Task, experimental setup

We consider a regression problem of a set of novels and their literary ratings. These ratings have been obtained in a large reader survey (about 14k participants),¹ in which 401 recent, bestselling Dutch novels (as well as works translated into Dutch) were rated on a 7-point Likert scale from *definitely not* to *highly literary*. The participants were presented with the author and title of each novel, and provided ratings for novels they had read. The ratings may have been influenced by well known authors or titles, but this does not affect the results of this paper because the machine learning models are not given such information. The task we consider is to predict the mean² rating for each novel. We ex-

¹The survey was part of The Riddle of Literary Quality, cf. <http://literaryquality.huygens.knaw.nl>

²Strictly speaking the Likert scale is ordinal and calls for the median, but the symmetric 7-point scale and the number of ratings arguably makes using the mean permissible; the latter provides more granularity and sensitivity to minority ratings.

clude 16 novels that have been rated by less than 50 participants. 91 % of the remaining novels have a *t*-distributed 95 % confidence interval < 0.5 ; e.g., given a mean of 3, the confidence interval typically ranges from 2.75 to 3.25. Therefore for our purposes the ratings form a reliable consensus. Novels rated as highly literary have smaller confidence intervals, i.e., show a stronger consensus. Where a binary distinction is needed, we call a rating of 5 or higher ‘literary.’

Since we aim to extract relevant features from the texts themselves and the number of novels is relatively small, we apply cross-validation, so as to exploit the data to the fullest extent while maintaining an out-of-sample approach. We divide the corpus in 5 folds of roughly equal size, with the following constraints: (a) novels by the same author must be in the same fold, since we want to rule out any influence of author style on feature selection or model validation; (b) the distribution of literary ratings in each fold should be similar to the overall distribution (stratification).

We control for length and potential particularities of the start of novels by considering sentences 1000–2000 of each novel. 18 novels with fewer than 2000 sentences are excluded. Together with the constraint of at least 50 ratings, this brings the total number of novels we consider to 369.

We evaluate the effectiveness of the features using a ridge regression model, with 5-fold cross-validation; we do not tune the regularization. The results are presented incrementally, to illustrate the contribution of each feature relative to the features before it. This makes it possible to gauge the effective contribution of each feature while taking any overlap into account.

We use R^2 as the evaluation metric, expressing the percentage of variance explained (perfect score 100); this shows the improvement of the predictions over a baseline model that always predicts the mean value (4.2, in this dataset). A mean baseline model is therefore defined to have an R^2 of 0. Other baseline models, e.g., always predicting 3.5 or 7, attain negative R^2 scores, since they perform worse than the mean baseline. Similarly, a random baseline will yield a negative expected R^2 .

3 Basic features

Sentence length, direct speech, vocabulary richness, and compressibility are simple yet effective stylistic features. We count direct speech sentences

by matching on specific punctuation; this provides a measure of the amount of dialogue versus narrative text in the novel. Vocabulary richness is defined as the proportion of words in a text that appear in the top 3000 most common words of a large reference corpus (Sonar 500; Oostdijk et al., 2013); this shows the proportion of difficult or unusual words. Compressibility is defined as the `bzip2` compression ratio of the texts; the intuition is that a repetitive and predictable text will be highly compressible. CLICHES is the number of cliché expressions in the texts based on an external dataset of 6641 clichés (van Wingerden and Hendriks, 2015); clichés, being marked as informal and unoriginal, are expected to be more prevalent in non-literary texts. Table 1 shows the results of these features. Several other features were also evaluated but were either not effective or did not achieve appreciable improvements when these basic features are taken into account; notably Flesch readability (Flesch, 1948), average dependency length (Gibson, 2000), and D-level (Covington et al., 2006).

	R^2
MEAN SENT. LEN.	16.4
+ % DIRECT SPEECH SENTENCES	23.1
+ TOP 3000 VOCAB.	23.5
+ BZIP2_RATIO	24.4
+ CLICHES	30.0

Table 1: Basic features, incremental scores.

4 Automatically induced features

In this section we consider extracting syntactic features, as well as three (sub)lexical baselines.

TOPICS is a set of 50 topic weights induced with Latent Dirichlet Allocation (LDA; Blei et al., 2003) from the corpus (for details, cf. Jautze et al., 2016).

Furthermore, we use character and word n -gram features. For words, bigrams present a good trade off in terms of informativeness (a bigram frequency is more specific than the frequency of an individual word) and sparsity (three or more consecutive words results in a large number of n -gram types with low frequencies). For character n -grams, $n = 4$ achieved good performance in previous work (e.g., Stamatatos, 2006).

We note three limitations of n -grams. First, the fixed n : larger or discontinuous chunks are not extracted. Combining n -grams does not help since a linear model cannot capture feature interactions, nor is the consecutive occurrence of two features

captured in the bag-of-words representation. Second, larger n imply a combinatorial explosion of possible features, which makes it desirable to select the most relevant features. Finally, word and character n -grams are surface features without linguistic abstraction. One way to overcome these limitations is to turn to syntactic parse trees and mine them for relevant features unrestricted in size.

Specifically, we consider tree fragments as features, which are arbitrarily-sized fragments of parse trees. If a parse tree is seen as consisting of a sequence of grammar productions, a tree fragment is a connected subsequence thereof. Compared to bag-of-word representations, tree fragments can capture both syntactic and lexical elements; and these combine to represent constructions with open slots (e.g., to take NP into account), or sentence templates (e.g., “Yes, but . . .”, he said). Tree fragments are thus a very rich source of features, and larger or more abstract features may prove to be more linguistically interpretable.

We present a data-driven method for extracting and selecting tree fragments. Due to combinatorics, there are an exponential number of possible fragments given a parse tree. For this reason it is not feasible to extract all fragments and select the relevant ones later; we therefore use a strategy to directly select fragments for which there is evidence of re-use by considering commonalities in pairs of trees. This is done by extracting the largest common syntactic fragments from pairs of trees (Sangati et al., 2010; van Cranenburgh, 2014). This method is related to tree-kernel methods (Collins and Duffy, 2002; Moschitti, 2006), with the difference that it extracts an explicit set of fragments. The feature selection approach is based on relevance and redundancy (Yu and Liu, 2004), similar to Swanson and Charniak (2013). Kim et al. (2011) also use tree fragments, for authorship attribution, but with a frequent tree mining approach; the difference with our approach is that we extract the largest fragments attested in each tree pair, which are not necessarily the most frequent.

4.1 Preprocessing

We parse the 369 novels with Alpino (Bouma et al., 2001). The parse trees include discontinuous constituents, non-terminal labels consist of both syntactic categories and function tags, selected morphological features,³ and constituents are bina-

³The DCOI tag set (van Eynde, 2005) is fine grained; we restrict the set to distinguish the 7 coarse POS tags, as well

alized head-outward with a markovization of $h=1$, $v=1$ (Klein and Manning, 2003).

For a fragment to be attested in a pair of parse trees, its labels need to match exactly, including the aforementioned categories, tags, and features. The $h = 1$ binarization implies that fragments may contain partial constituents; i.e., a contiguous sequence of children from an n -ary constituent.

Figure 1 shows an example parse tree; for brevity, this tree is rendered without binarization. The non-terminal labels consist of a syntactic category (shown in red), followed by a function tag (green). The part-of-speech tags additionally have morphological features (black) in square brackets. Some labels contain percolated morphological features, prefixed by a colon.

4.2 Mining syntactic tree fragments

The procedure is divided in two parts. The first part concerns fragment extraction:

1. Given texts divided in folds $F_1 \dots F_n$, each C_i is the set of parse trees obtained from parsing all texts in F_i . Extract the largest common fragments of the parse trees in all pairs of folds $\langle C_i, C_j \rangle$ with $i < j$. A common fragment f of parse trees t_1, t_2 is a connected subgraph of t_1 and t_2 . The result is a set of initial candidates that occur in at least two different texts, stored separately for each pair of folds $\langle C_i, C_j \rangle$.
2. Count occurrences of all fragments in all texts.

Fragment selection is done separately w.r.t. each test fold. Given test fold i , we consider the fragments found in training folds $\{1..n\} \setminus i$; e.g., given $n = 5$, for test fold 1 we select only from the fragments and their counts as observed in training folds 2–5. Given a set of fragments from training folds, selection proceeds as follows:

1. Zero count threshold: remove fragments that occur in less than 5 % of texts (too specific to particular novels); frequency threshold: remove fragments that occur less than 50 times across the corpus (too rare to reliably detect a correlation with the ratings).
2. Relevance threshold: select fragments by considering the correlation of their counts with the literary ratings of the novels in the training folds. Apply a simple linear regression

as infinite verbs, auxiliary verbs, proper nouns, subordinating conjunctions, personal pronouns, and postpositions.

based on the Pearson correlation coefficient, and use an F-test to filter out fragments whose p -value⁴ > 0.05 . The F-test determines significance based on the number of datapoints N , and the correlation r ; the effective threshold is approximately $|r| > 0.11$.

3. Redundancy removal: greedily select the most relevant fragment and remove other fragments that are too similar to it. Similarity is measured by computing the correlation coefficient between the feature vectors of two fragments, with a cutoff of $|r| > 0.5$. Experiments where this step was not applied indicated that it improves performance.

Note that there is some risk of overfitting since fragments are both extracted and selected from the training set. However, this is mitigated by the fact that fragments are extracted from pairs of folds, while selection is constrained to fragments that are attested and significantly correlated across the whole training set.

The values for the thresholds were chosen manually and not tuned, since the limited number of novels is not enough to provide a proper tuning set. Table 2 lists the number of fragments extracted from folds 2–5 after each of these steps.

recurring fragments	3,193,952
occurs in $> 5\%$ of texts	375,514
total freq. > 50 across corpus	98,286
relevance: correlated s.t. $p < 0.05$	30,044
redundancy: $ r < 0.5$	7,642

Table 2: The number of fragments in folds 2–5 after each filtering step.

4.3 Evaluation

Due to the large number of induced features, Support Vector Regression (SVR) is more effective than ridge regression. We therefore train a linear SVR model with the same cross-validation setup, and feed its predictions to the ridge regression model (i.e., stacking). Feature counts are turned into relative frequencies. The model has two hyperparameters: C determines the regularization, and ϵ is a threshold beyond which predictions are considered good enough during training. Instead of

⁴If we were actually testing hypotheses we would need to apply Bonferroni correction to avoid the Family-Wise Error due to multiple comparisons; however, since the regression here is only a means to an end, we leave the p -values uncorrected.

	1	2	3	4	5	Mean
Word Bigrams	59.8	47.0	58.0	63.6	50.7	55.8
Char. 4-grams	58.6	50.4	54.2	65.0	56.2	56.9
Fragments	61.6	53.4	58.7	65.8	46.5	57.2

Table 3: Regression evaluation. R^2 scores on the 5 cross-validation folds.

	R^2
BASIC FEATURES (TABLE 1)	30.0
+ TOPICS	52.2
+ BIGRAMS	59.5
+ CHAR. 4-GRAMS	59.9
+ FRAGMENTS	61.2

Table 4: Automatically induced features; incremental scores.

tuning these parameters we pick fixed values of $C=100$ and $\epsilon=0$, reducing regularization compared to the default of $C=1$ and disabling the threshold.

Cf. Table 3 for the scores. The syntactic fragments perform best, followed by char. 4-grams and word bigrams. We report scores for each of the 5 folds separately because the variance between folds is high. However, the differences between the feature types are relatively consistent. The variance is not caused by the distribution of ratings, since the folds were stratified on this. Nor can it be explained by the agreement in ratings per novel, since the 95 % confidence intervals of the individual ratings for each novel were of comparable width across the folds. Lastly, author gender, genre, and whether the novel was translated do not differ markedly across the folds. It seems most likely that the novels simply differ in how predictable their ratings are from textual features.

In order to gauge to what extent these automatically induced features are complementary, we combine them in a single model together with the basic features; cf. the scores in Table 4. Both character 4-grams and syntactic fragments still provide a relatively large improvement over the previous features, taking into account the inherent diminishing returns of adding more features.

Figure 2 shows a bar plot of the ten novels with the largest prediction error with the fragment and word bigram models. Of these novels, 9 are highly literary and underestimated by the model. For the other novel (Smeets, *Afrekening*) the literary rating is overestimated by the model. Since this top 10 is based on the mean prediction from both models, the error is large for both models. This does not

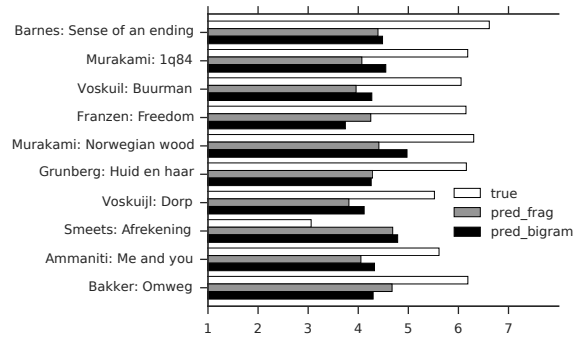


Figure 2: The ten novels with the largest prediction error (using both fragments and bigrams).

Novel	residual (true - pred.)	mean sent. len.	% direct speech	% Top 3000 vocab.	bzip2 ratio
Rosenboom: Zoete mond	0.075	23.5	24.7	0.80	0.31
Mortier: Godenslaap	0.705	24.9	25.2	0.77	0.34
Lewinsky: Johannistag	0.100	18.3	28.6	0.85	0.32
Eco: The Prague cemetery	0.148	24.5	15.7	0.79	0.33
Franzen: Freedom	2.154	16.2	56.8	0.84	0.33
Barnes: Sense of an ending	2.143	14.1	23.1	0.85	0.32
Voskuil: Buurman	2.117	7.66	58.0	0.89	0.28
Murakami: 1q84	1.870	12.3	20.4	0.84	0.32

Table 5: Comparison of baseline features for novels with good (1–4) and bad (5–8) predictions.

change when the top 10 errors using only fragments or bigrams is inspected; i.e., the hardest novels to predict are hard with both feature types.

What could explain these errors? At first sight, there is no obvious commonality between the literary novels that are predicted well, or between the ones with a large error; e.g., whether the novels have been translated or not does not explain the error. A possible explanation is that the successfully predicted literary novels share a particular (e.g., rich) writing style that sets them apart from other novels, while the literary novels that are underestimated by the model are not marked by such a writing style. It is difficult to confirm this directly by inspecting the model, since each prediction is the sum of several thousand features, and the contributions of these features form a long tail. If we define the contribution of a feature as the absolute value of its weight times its relative frequency in the document, then in case of Barnes, *The sense of an ending*, the top 100 features contribute only 34 % of the total prediction.

Table 5 gives the basic features for the top 4 literary novels with the largest error and contrasts them with 4 literary novels which are well predicted. The most striking difference is sentence length: the underestimated literary novels have

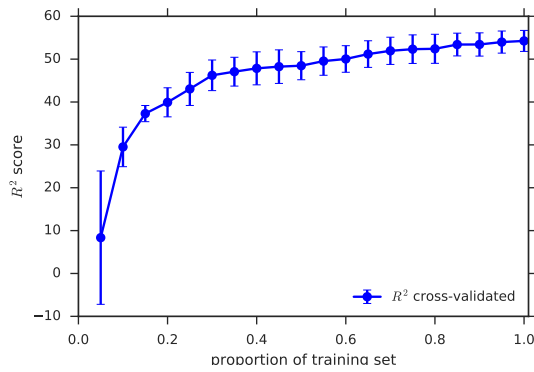


Figure 3: Learning curve when varying training set size. The error bars show the standard error.

markedly shorter sentences. Voskuil and Franzen have a higher proportion of direct speech (they are in fact the only literary novels in the top 10 novels with the most direct speech). Lastly, the underestimated novels have a higher proportion of common words (lower vocabulary richness). These observations are compatible with the explanation suggested above, that a subset of the literary novels share a simple, readable writing style with non-literary novels. Such a style may be more difficult to detect than a literary style with long and complex sentences, or rich vocabulary and phraseology, because a simple, well-crafted sentence may not offer overt surface markers of stylization. Book reviews appear to support this notion for *The sense of an ending*: “A slow burn, measured but suspenseful, this compact novel makes every slyly crafted sentence count” (Tonkin, 2011); and “polished phrasings, elegant verbal exactness and epigrammatic perceptions” (Kemp, 2011).

In order to test whether the amount of data is sufficient to learn to predict the ratings, we construct a learning curve for different training set sizes; cf. Figure 3. The set of novels is shuffled once, so that initial segments of different size represent random samples. The novels are sampled in 5 % increments (i.e., 20 models are trained). The graphs show the cross-validated scores.

The graphs show that increasing the number of novels has a large effect on performance. The curve is steep up to 30 % of the training set, and the performance keeps improving steadily but more slowly up to the last data point. Since the performance is relatively flat starting from 85 %, we can conclude that the k -fold cross-validation with $k = 5$ provides an adequate estimate of the model’s performance if

	R^2
BASIC FEATURES (TABLE 1)	30.0
+ AUTO. INDUCED FEAT. (TABLE 4)	61.2
+ GENRE	74.3
+ TRANSLATED	74.0
+ AUTHOR GENDER	76.0

Table 6: Metadata features; incremental scores.

it were trained on the full dataset; if the model was still gaining performance significantly with more training data, the cross-validation score would underestimate the true prediction performance.

A similar experiment was performed varying the number of features. Here the performance plateaus quickly and reaches an R^2 of 53.0 % at 40 %, and grows only slightly from that point.

5 Metadata features

In addition to textual features, we also include three (categorical) metadata features not extracted from the text, but still an inherent feature of the novel in question: GENRE, TRANSLATED, and AUTHOR GENDER; cf. Table 6 for the results. Figure 4 shows a visualization of the predictions in a scatter plot.

GENRE is the coarse genre classification Fiction, Suspense, Romantic, Other, derived from the publisher’s categorization. Genre alone is already a strong predictor, with an R^2 of 58.3 on its own. However, this score is arguably misleading, because the predictions are very coarse due to the discrete nature of the feature.

A striking result is that the variables AUTHOR GENDER and TRANSLATED increase the score, but only when they are both present. Inspecting the mean ratings shows that translated novels by female authors have an average rating of 3.8, while originally Dutch male authors are rated 5.0 on average; the ratings of the other combinations lie in between these extremes. This explains why the combination works better than either feature on its own, but due to possible biases inherent in the makeup of the corpus, such as which female or translated authors are published and selected for the corpus, no conclusions on the influence of gender or translation should be drawn from these datapoints.

6 Previous work

Table 7 shows an overview of previous work on the task of predicting the (literary) quality of novels. Note that the datasets and targets differ, therefore none of the results are directly comparable. For

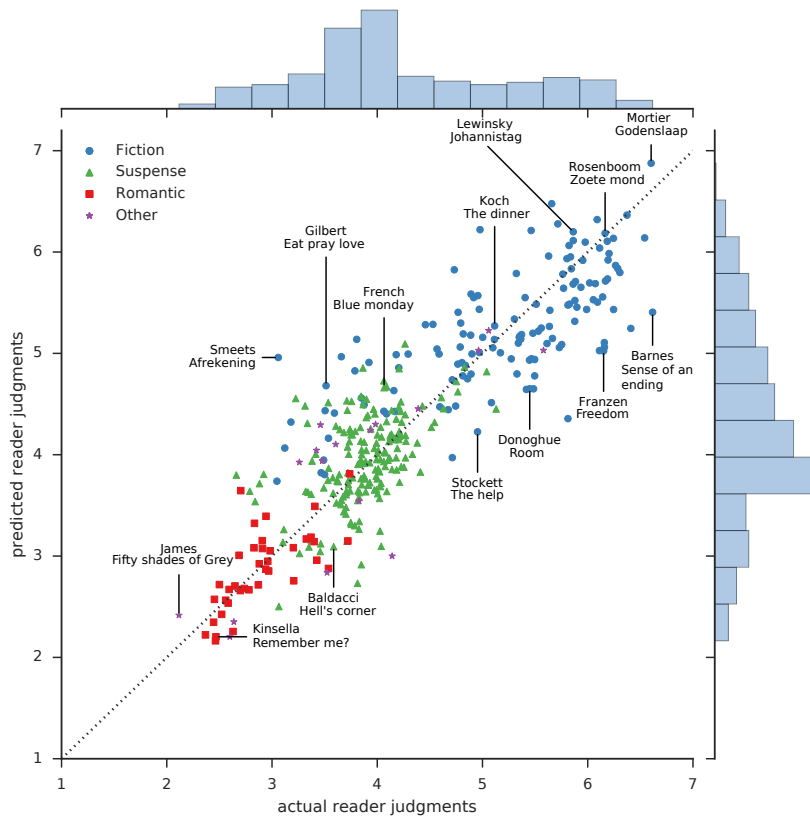


Figure 4: A scatter plot of regression predictions and actual literary ratings. Original/translated titles. Note the histograms beside the axes showing the distribution of ratings (top) and predictions (right).

example, regression is a more difficult task than binary classification, and recognizing the difference between an average and highly literary novel is more difficult than distinguishing either from a different domain or genre (e.g., newswire).

Louwerse et al. (2008) discriminate literature from other texts using Latent Semantic Analysis. Ashok et al. (2013) use bigrams, POS tags, and grammar productions to predict the popularity of Gutenberg texts. van Cranenburgh and Koolen (2015) predict the literary ratings of texts, as in the present paper, but only using bigrams, and on a smaller, less diverse corpus. Compared to previous work, this paper gives a more precise estimate of how well shades of literariness can be predicted from a diverse range of features, including larger and more abstract syntactic constructions.

7 Analysis of selected tree fragments

An advantage of parse tree fragments is that they offer opportunities for interpretation in terms of linguistic aspects as well as basic distributional aspects such as shape and size.

Figure 5 shows three fragments ranked highly

Binary classification	Dataset, task	Acc.
Louwerse et al. (2008)	119 all-time literary classics and 55 other texts, literary novels vs. non-fiction/sci-fi	87.4
Ashok et al. (2013)	800 19th century novels, low vs. high download count	75.7
van Cranenburgh and Koolen (2015)	146 recent novels, low vs. high survey ratings	90.4
Regression result	Dataset, task	R^2
van Cranenburgh and Koolen (2015)	146 recent novels, survey ratings	61.3
This work	401 recent novels, survey ratings	76.0

Table 7: Overview of previous work on modeling (literary) quality of novels.

by the correlation metric, as extracted from the first fold. The first fragment shows an incomplete constituent, indicated by the ellipses as first and last leaves. Such incomplete fragments are made possible by the binarization scheme (cf. Sec. 4.1).

Table 8 shows a breakdown of fragment types in the first fold. In contrast with n -grams, we also see

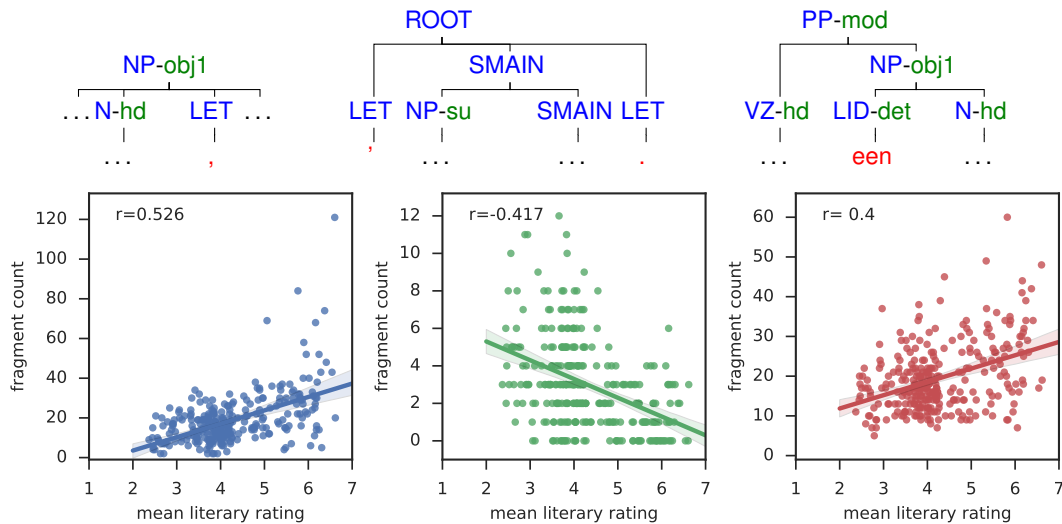


Figure 5: Three fragments whose frequencies in the first fold have a high correlation with the literary ratings. Note the different scales on the y -axis. From left to right; Blue: complex NP with comma; Green: quoted speech; Red: Adjunct PP with indefinite article.

fully lexicalized	1,321
syntactic (no lexical items)	2,283
mixed	4,038
<hr/>	
discontinuous	684
discontinuous substitution site	396
<hr/>	
total	7,642

Table 8: Breakdown of fragment types selected in the first fold.

a large proportion of purely syntactic fragments, and fragments mixing both lexical elements and substitution sites. In the case of discontinuous fragments, it turns out that the majority has a positive correlation; this might be due to being associated with more complex constructions.

Figure 6 shows a breakdown by fragment size (defined as number of non-terminals), distinguishing fragments that are positively versus negatively correlated with the literary ratings.

Note that 1 and 3 are special cases corresponding to lexical (e.g., DT \rightarrow the) and binary grammar productions (e.g., NP \rightarrow DT N), respectively. The fragments with 2, 4, and 6 non-terminals are not as common because an even number implies the presence of unary nodes. Except for fragments of size 1, the frontier of fragments can consist of either substitution sites or terminals (since we distinguish only the number of non-terminals). On the one hand smaller fragments corresponding to one or two grammar productions are most common, and are predominantly positively correlated with the

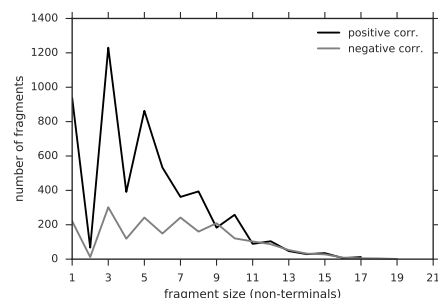


Figure 6: Breakdown by fragment size (number of non-terminals).

literary ratings. On the other hand there is a significant negative correlation between fragment size and literary ratings ($r = -0.2, p < 0.001$); i.e., smaller fragments tend to be positively correlated with the literary ratings.

It is striking that there are more positively than negatively correlated fragments, while literary novels are a minority in the corpus (88 out of 369 novels are rated 5 or higher). Additionally, the breakdown by size shows that the larger number of positively correlated fragments is due to a large number of small fragments of size 3 and 5; however, combinatorially, the number of possible fragment types grows exponentially with size (as reflected in the initial set of recurring fragments), so larger fragment types would be expected to be more numerous. In effect, the selected negatively correlated fragments ignore this distribution by being relatively uniform with respect to size, while the

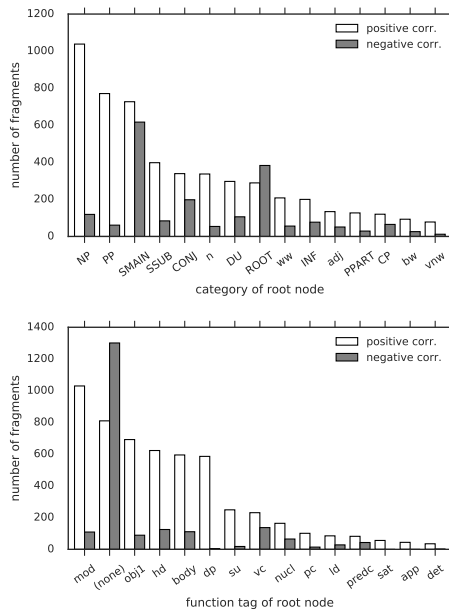


Figure 7: Breakdown by category (above) and function tag (below) of fragment root (top 15 labels).

literary fragments actually show the opposite distribution.

What could explain the peak of positively correlated, small fragments? In order to investigate the peak of small fragments, we inspect the 40 fragments of size 3 with the highest correlations. These fragments contain indicators of unusual or more complex sentence structure:

- DU, dp: discourse phenomena for which no specific relation can be assigned (e.g., discourse relations beyond the sentence level).
- appositive NPs, e.g., ‘John the artist.’
- a complex NP, e.g., containing punctuation, nested NPs, or PPs.
- an NP containing an adjective used nominally or an infinitive verb.

On the other hand, most non-literary fragments are top-level productions containing ROOT or clause-level labels, for example to introduce direct speech.

Another way of analyzing the selected fragments is by frequency. When we consider the total frequencies of selected fragments across the corpus, there is a range of 50 to 107,270. The bulk of fragments have a low frequency (before fragment selection 2 is by far the most dominant frequency), but the tail is very long. Except for the fact that there is a larger number of positively correlated fragments, the histograms have a very similar shape.

Lastly, Figure 7 shows a breakdown by the syn-

tactic categories and function tags of the root node of the fragments. The positively correlated fragments are spread over a larger variety of both syntactic categories and function tags. This means that for most labels, the number of positively correlated fragments is higher; the exceptions are ROOT, SV1 (a verb-initial phrase, not part of the top 15), and the absence of a function tag (indicative of a non-terminal directly under the root node). All of these exceptions point to a tendency for negatively correlated fragments to represent templates of complete sentences.

8 Conclusion

The answer to the main research question is that literary judgments are non-arbitrary and can be explained to a large extent from the text itself: there is an intrinsic *literariness* to literary texts. Our model employs an ensemble of textual features that show a cumulative improvement on predictions, achieving a total score of 76.0 % variation explained. This result is remarkably robust: not just broad genre distinctions, but also finer distinctions in the ratings are predicted.

The experiments showed one clear pattern: literary language tends to use a larger set of syntactic constructions than the language of non-literary novels. This also provides evidence for the hypothesis that literature employs a specific inventory of constructions. All evidence points to a notion of literature which to a substantial extent can be explained purely from internal, textual factors, rather than being determined by external, social factors.

Code and details of the experimental setup are available at <https://github.com/andreasvc/literariness>

Acknowledgments

We are grateful to David Hoover, Patrick Juola, Corina Koolen, Laura Kallmeyer, and the reviewers for feedback. This work is part of The Riddle of Literary Quality, a project supported by the Royal Netherlands Academy of Arts and Sciences through the Computational Humanities Program. In addition, part of the work on this paper was funded by the German Research Foundation DFG (*Deutsche Forschungsgemeinschaft*).

References

- Vikas Ashok, Song Feng, and Yejin Choi. 2013. Success with style: using writing style to predict the success of novels. In *Proceedings of EMNLP*, pages 1753–1764. <http://aclweb.org/anthology/D13-1181>.
- Chris Baldick. 2008. Literariness. In *The Oxford Dictionary of Literary Terms*. Oxford University Press, USA.
- Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proceedings of NAACL*, pages 327–337. <http://aclweb.org/anthology/N12-1033>.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022. <http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>.
- Rens Bod. 1992. A computational model of language performance: Data-oriented parsing. In *Proceedings COLING*, pages 855–859. <http://aclweb.org/anthology/C92-3126>.
- Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. *Language and Computers*, 37(1):45–59. <http://www.let.rug.nl/vannoord/papers/alpino.pdf>.
- Pierre Bourdieu. 1996. *The rules of art: Genesis and structure of the literary field*. Stanford University Press.
- Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL*. <http://aclweb.org/anthology/P02-1034>.
- Michael A. Covington, Congzhou He, Cati Brown, Lorina Naci, and John Brown. 2006. How complex is that sentence? A proposed revision of the Rosenberg and Abbeduto D-Level scale. CASPR Research Report 2006-01, Artificial Intelligence Center.
- Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.
- Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, pages 95–126.
- Wendell V. Harris. 1995. *Literary meaning. Reclaiming the study of literature*. Palgrave Macmillan, London.
- Kim Jautze, Andreas van Cranenburgh, and Corina Koolen. 2016. Topic modeling literary quality. In *Digital Humanities 2016: Conference Abstracts*, pages 233–237, Kraków, Poland. <http://dh2016.adho.org/abstracts/95>.
- Peter Kemp. 2011. The sense of an ending by Julian Barnes. Book review, *The Sunday Times*, July 24. <http://www.thesundaytimes.co.uk/sto/culture/books/fiction/article674085.ece>.
- Sangkyum Kim, Hyungsul Kim, Tim Weninger, and Jiawei Han. 2011. Authorship classification: A syntactic tree mining approach. In *Proceedings of SIGIR*, pages 455–464. ACM. <http://dx.doi.org/10.1145/2009916.2009979>.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, volume 1, pages 423–430. <http://aclweb.org/anthology/P03-1054>.
- Max Louwerse, Nick Benesh, and Bin Zhang. 2008. Computationally discriminating literary from non-literary texts. In S. Zyngier, M. Bortolussi, A. Chesnokova, and J. Auracher, editors, *Directions in empirical literary studies: In honor of Willie Van Peer*, pages 175–191. John Benjamins Publishing Company, Amsterdam.
- Ronan McDonald. 2007. *The death of the critic*. Continuum, London.
- Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of EACL*, pages 113–120. <http://aclweb.org/anthology/E06-1015>.
- Nelleke Oostdijk, Martin Reynaert, Véronique Hoste, and Ineke Schuurman. 2013. The construction of a 500-million-word reference corpus of contemporary written dutch. In *Essential speech and language technology for Dutch*, pages 219–247. Springer.
- Federico Sangati, Willem Zuidema, and Rens Bod. 2010. Efficiently extract recurring tree fragments from large treebanks. In *Proceedings of LREC*, pages 219–226. <http://dare.uva.nl/record/371504>.
- Remko Scha. 1990. Language theory and language technology; competence and performance. In Q.A.M. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere, the Netherlands. Original title: Taaltheorie en taaltechnologie; competence en performance. English translation: <http://iaaa.nl/rs/LeerdamE.html>.
- Efstathios Stamatatos. 2006. Ensemble-based author identification using character n-grams. In *Proceedings of the 3rd International Workshop on Text-based Information Retrieval*, pages 41–46. <http://ceur-ws.org/Vol-205/paper8.pdf>.
- Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556. <http://dx.doi.org/10.1002/asi.21001>.

- Benjamin Swanson and Eugene Charniak. 2012. Native language detection with tree substitution grammars. In *Proceedings of ACL*, pages 193–197. <http://aclweb.org/anthology/P12-2038>.
- Ben Swanson and Eugene Charniak. 2013. Extracting the native language signal for second language acquisition. In *Proceedings of NAACL-HLT*, pages 85–94. <http://aclweb.org/anthology/N13-1009>.
- Boyd Tonkin. 2011. The sense of an ending, by Julian Barnes. Book review, *The Independent*, August 5. <http://www.independent.co.uk/arts-entertainment/books/reviews/the-sense-of-an-ending-by-julian-barnes-2331767.html>.
- Andreas van Cranenburgh and Corina Koolen. 2015. Identifying literary texts with bigrams. In *Proc. of workshop Computational Linguistics for Literature*, pages 58–67. <http://aclweb.org/anthology/W15-0707>.
- Andreas van Cranenburgh. 2012. Literary authorship attribution with phrase-structure fragments. In *Proceedings of CLFL*, pages 59–63. Revised version: <http://andreasvc.github.io/clfl2012.pdf>.
- Andreas van Cranenburgh. 2014. Extraction of phrase-structure fragments with a linear average time tree kernel. *Computational Linguistics in the Netherlands Journal*, 4:3–16. <http://www.clinjournal.org/sites/default/files/01-Cranenburgh-CLIN2014.pdf>.
- Frank van Eynde. 2005. Part of speech tagging en lemmatisering van het D-COI corpus. Transl.: *POS tagging and lemmatization of the D-COI corpus*, tech report, July 2005. <http://www.ccl.kuleuven.ac.be/Papers/DCOIpos.pdf>.
- Wouter van Wingerden and Pepijn Hendriks. 2015. *Dat Hoor Je Mij Niet Zeggen: De allerbeste taalclichés*. Thomas Rap, Amsterdam. Transl.: You didn't hear me say that: The very best linguistic clichés.
- Lei Yu and Huan Liu. 2004. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224. <http://jmlr.org/papers/volume5/yu04a/yu04a.pdf>.

Aye or naw, whit dae ye hink?

Scottish independence and linguistic identity on social media

Philippa Shoemark*

p.j.shoemark@ed.ac.uk

Debnil Sur[†]

debnil@stanford.edu

Luke Shrimpton*

luke.shrimpton@ed.ac.uk

Iain Murray*

i.murray@ed.ac.uk

Sharon Goldwater*

sgwater@inf.ed.ac.uk

*School of Informatics
University of Edinburgh

[†]Department of Computer Science
Stanford University

Abstract

Political surveys have indicated a relationship between a sense of Scottish identity and voting decisions in the 2014 Scottish Independence Referendum. Identity is often reflected in language use, suggesting the intuitive hypothesis that individuals who support Scottish independence are more likely to use distinctively Scottish words than those who oppose it. In the first large-scale study of sociolinguistic variation on social media in the UK, we identify distinctively Scottish terms in a data-driven way, and find that these terms are indeed used at a higher rate by users of pro-independence hashtags than by users of anti-independence hashtags. However, we also find that in general people are *less* likely to use distinctively Scottish words in tweets with referendum-related hashtags than in their general Twitter activity. We attribute this difference to style-shifting relative to audience, aligning with previous work showing that Twitter users tend to use fewer local variants when addressing a broader audience.

1 Introduction

A central idea from sociolinguistics is that people's social identity is reflected in their use of language, and that people modulate their use of language in order to present particular identities in different situations. The recent availability of social media data has raised interest in confirming and extending these results using large scale datasets. For example, Twitter data has been used to examine patterns

of regional variation in general US English (Doyle, 2014; Huang et al., 2015), African American English (Jones, 2015), and global Spanish (Gonçalves and Sánchez, 2014), and to study variation associated with factors such as race/ethnicity (Jones, 2015; Blodgett et al., 2016; Jørgensen et al., 2015) and gender (Bamman et al., 2014). These studies have shown that tweets mirror spoken language in many ways, such as displaying dialect variation not only in the use of distinct lexical items, but also in the use of non-standard spellings to indicate non-standard pronunciation—in fact, these spellings even reflect the phonological processes found in spoken language (Eisenstein, 2015). There is also evidence that, as in spoken language, individuals may shift their style of language in response to the audience. In particular, studies have found that when the expected audience of a tweet is larger, Americans use fewer non-standard and local words (Pavalanathan and Eisenstein, 2015) and Dutch bilingual speakers of a minority language are more likely to use Dutch rather than their other language (Nguyen et al., 2015). A small-scale case study of a single Scottish Twitter user also provides preliminary evidence that users may modulate their production of regional variants according to the topic of the tweet (Tatman, 2015).

Here we present the first large-scale sociolinguistic study of British tweets, and the first to examine the relationship between sociolinguistic variation and political views using social media data. We use a large corpus of tweets to examine the relationship between users' linguistic choices and their views about the 2014 Scottish independence referendum. The referendum (on whether Scotland should leave the UK) generated considerable political discussion and an unprecedented turnout of 84.6% of the

electorate, with the ‘No’ (anti-independence) side taking 55.3% of the vote. The 2013 Scottish Social Attitudes Survey (ScotCen, 2013) showed a clear correlation between national identity and voting intentions (53% of those who identified as ‘Scottish not British’ said they intended to vote ‘Yes’ to independence, vs. just 5% of those who identified as ‘British not Scottish’), and there was much discussion in the popular press about the relationship between a sense of Scottish identity and support for Scottish sovereignty.

Although this recent discussion was not centered on language, there is a long history of scholarly discourse connecting the use of the Scots language¹ and sociolinguistic and political identity (Grant, 1931; McAfee, 1985; Corbett et al., 2003). If this connection still holds today, then we might expect to find that those on the ‘Yes’ side of the debate use more identifiably Scottish language than those on the ‘No’ side. We might also expect to find some modulation of Scottish language use depending on whether users are discussing the referendum or not.

To examine these questions, we used a data-driven approach to identify linguistic terms that are used more in Scotland than in the rest of the UK. The identified terms include uniquely Scots words that are attested in Scots literature dating back to the 1600s and earlier, contemporary regional colloquialisms, spelling variants of Standard English words which reflect Scottish pronunciations, and acronyms used as shorthand for distinctive Scottish phrases. From these, we selected variables for which users can produce either a Standard English or Scottish variant (e.g., DO vs. DAE). We then classified users as pro- or anti-independence based on the referendum-related hashtags they used and asked whether these two groups use Scottish variants at different rates. We found that the pro-independence group did use Scottish variants significantly more than the anti-independence group, although the overall rate of Scottish variants is very low amongst all users.

Next, we compared the use of Scottish variants in tweets containing referendum-related hashtags to their use in other tweets. If users are aiming to project their Scottish identity as part of politi-

cal discourse, then we might expect greater use of Scottish variants in referendum tweets than in non-referendum tweets. However, previous studies have suggested that non-standard and local variants are used *less* frequently in tweets containing hashtags, which typically have a larger audience than other tweets (Pavalanathan and Eisenstein, 2015). This effect would predict the opposite result—a lower use of Scottish variants in tweets with referendum hashtags—and indeed this is the result we found. So it appears that although pro-independence users do make greater use of Scottish variants overall, they do not increase their Scottish usage when engaging in broad-audience political discourse.

To summarize, the contributions of our paper are: (1) The first large-scale study of dialect variation on twitter in the UK. We show that in addition to using Scots in speech and some literary genres such as poetry, people are using Scots in informal public writing. The data-driven approach enables us to identify Scotland-specific lexical items without relying on pre-conceived notions of which variables to look for (cf. Tatman, 2015), and reveals that in addition to using attested Scots vocabulary, Twitter users appear to be creatively adapting to the medium with their use of acronyms for distinctly Scottish turns of phrase. (2) The first study connecting sociolinguistic variables to political stance using social media data, showing that pro-independence users have a higher rate of Scottish usage. (3) Further evidence of Pavalanathan and Eisenstein’s (2015) claim that Twitter users modulate their language according to the audience, with local variants being less likely in tweets directed to larger audiences.

2 Context

‘Scots’ refers to the group of dialects historically spoken in the Lowlands of Scotland. While Scots has Anglo-Scandinavian origins in common with English, by the 16th century its pronunciation, vocabulary, and literary norms had considerably diverged from those of English, and Scots had become established as the prestige language in Scotland (Kay, 1988).² However, following the Union of Crowns in 1603, when King James VI of Scotland acceded to the thrones of England and Ireland,

¹Historically, Scots has been considered a different language than English (see §2), though with many cognates and overlapping vocabulary. Most native Scottish people today speak some variety of Scottish English, which retains a few uniquely Scots words but is mainly distinguished from other varieties of English by its pronunciation.

²Previously, Gaelic had been the dominant spoken and literary language in Scotland. Note that while in medieval times non-Gaelic speakers referred to the Gaels as ‘Scots’, what we now refer to as ‘Scots’ is the Anglo-Scandinavian language which spread at the expense of Scottish Gaelic (a Celtic language) in the 15th & 16th centuries.

he and his court began to adopt English norms in their writing. After the Union of Parliaments in 1707, English firmly replaced Scots as the language of serious or elevated discourse in Scotland (Grant, 1931). While some people still use distinctive elements of Scots in their speech, until recently the average Scottish person’s exposure to written Scots would have been largely confined to a select few literary domains such as poetry and comic narrative (Corbett et al., 2003). However, social media has given rise to a new genre of casual, communicative writing that is potentially visible to large and diverse audiences, providing both a platform and an impetus to express one’s identity through the use of written language. Below, we provide three example tweets (each from a different user) which contain orthographic representations of Scots vocabulary and/or Scottish English pronunciation. Standard English variants of Scottish terms are provided in italics.

- (1) No matter how shite [*shit*] a day you’ve had just remember there’s always good biscuits in yer [*your*] grannies hoose [*house*]
- (2) “Absolute carnage” at polling station earlier. Bairns [*kids*] playing, polite grannies, Yessers and Nos blethering [*blathering*] to each other. #VoteYesScotland
- (3) #fuckoffscotland hud on we will fuck off but afore we dae eh challenge ye tae a square go ya queen loving DIDDY doughnut Sasijs YUP-TAE
#fuckoffscotland hold on we will fuck off but before we do I challenge you to a fair fight you queen loving fools. What are you doing!?

3 Data

Our data was drawn from the Sample endpoint of Twitter’s Streaming API (a.k.a. the ‘Spritzer’), which provides a random 1% sample of all public tweets in near real-time. We started with all tweets streamed from the Spritzer between 1st September 2013 and 30th September 2014. These dates cover a year of activity leading up to the referendum, as well as the day the vote took place (18 September 2014), and immediate reactions. We used a language classifier (Lui and Baldwin, 2012) to filter out non-English tweets, yielding an initial dataset of 629,431,509 tweets.³ Because we are interested

³One might be concerned that an automatic language filter could remove some of the heavily Scottish tweets. However,

in the linguistic choices that individuals make in various contexts, we took steps to remove tweets which were not originally authored by the individual who posted them. Retweets (tweets which are verbatim copies of other tweets) were identified by a case-insensitive search for the token ‘RT’, and discarded. Quote tweets (tweets which contain verbatim copies of other tweets, but are augmented with original comments) were dealt with by discarding any text between double quotation marks, but retaining the remainder of the tweet.

From this initial dataset we extracted three overlapping subsets:

The Geotagged-UK (GU) dataset contains all tweets geotagged to a location in the United Kingdom (1,654,204 tweets by 446,923 distinct users).

The Geotagged-Scotland (GS) dataset contains all tweets geotagged to a location in Scotland (166,992 tweets by 40,861 distinct users).

The Indyref Tweets (IT) dataset consists of tweets containing hashtags relating to the 2014 Scottish Independence Referendum.

To construct the IT dataset, we first created a list of relevant hashtags, starting with the following five seed hashtags: #IndyRef, #VoteYes, #VoteNo, #YesScotland, #BetterTogether.⁴ For each of these five seeds, we extracted from our initial filtered dataset a list of *all* tweets by any user who used the seed hashtag. We identified the 100 most frequent hashtags in each of these five lists of tweets, and manually discarded all hashtags which were unrelated to the referendum, as well as those which were highly ambiguous (e.g., #Indy, which sometimes refers to the referendum, but also commonly refers to a genre of music). The resulting list of referendum-related hashtags is given in Table 1.

Next, we extracted all tweets from our initial dataset which contain at least one of the hashtags on this list, yielding 77,708 tweets by 26,019 distinct users. We then applied a heuristic to filter out tweets produced by bots and spammers: for

even tweets such as example (3) in §2 are assigned a very high probability of being English by the filter. Perhaps other tweets with many Scottish terms were filtered out, in which case we will underestimate the probability that users choose Scottish variants. However this issue should not cause us to find differences in use between different groups where there are none.

⁴‘Yes Scotland’ and ‘Better Together’ are the names of the principal organisations representing the Yes and No vote campaigns, respectively.

each user in the IT dataset for whom we had at least 5 tweets in the initial dataset, we computed the proportion of their tweets that contain URLs, and discarded users for whom this proportion was in the 90th percentile. This step filtered out 11,443 tweets by 1389 users.

Note that seven of the hashtags in Table 1 (*#voteyes*, *#bettertogether*, *#nothanks*, *#voteno*, *#yes2014*, *#letsstaytogether*, and *#yesvote*) are occasionally used in contexts unrelated to the Scottish Independence Referendum (e.g. *#bettertogether* can also refer to interpersonal relationships). However, they are distinctive enough that if a user has also used hashtags which are unambiguously related to the referendum, then it seems reasonable to assume that their usage of these potentially-ambiguous hashtags relates to the referendum too. Therefore, in order for a tweet containing one of these seven hashtags to be retained in the Indyref dataset, we required that its author had also used at least one other hashtag from Table 1. This criterion filtered out a further 6601 tweets by 6041 distinct users, such that the final IT dataset contains 59,664 tweets by 18,589 distinct users.

4 Identifying distinctively Scottish vocabulary on Twitter

We wish to identify terms that are more likely to be used by Twitter users in Scotland than in the rest of the UK. We follow the method of Pavalanathan and Eisenstein (2015), who used the Sparse Additive Generative Model of Text (SAGE) framework (Eisenstein et al., 2011) to identify tweet terms associated with metropolitan areas in the United States. SAGE models deviations in the log-frequencies of terms in a corpus of interest (here, the GS dataset) with respect to their log-frequencies in some “background” corpus (here, the GU dataset). The estimated deviations are regularized to avoid overstating the importance of deviations in the frequencies of rare words. Here, we use a publicly available implementation of SAGE⁵ to obtain log-frequency deviation estimates for all terms which occur at least fifty times in the GU dataset, excluding hashtags, mentions, URLs, and stopwords. The terms with the highest estimates are those which are most distinctive to tweets geo-located in Scotland.

⁵<https://github.com/jacobeisenstein/jos-gender-2014/>

4.1 Scotland-specific terms

Unsurprisingly, many of the Scotland-specific terms are proper nouns which are topically associated with Scotland, such as Scottish placenames, political figures, and sports personalities. There are also several common nouns (e.g. ‘devolution’, ‘bagpipes’) and verbs (e.g. ‘canvass’, ‘invade’) which are strongly associated with the political or cultural climate in Scotland. These terms occur with greater relative frequency in the GS dataset simply because their referents are discussed with greater relative frequency; not because they are distinct from the terms that people in the rest of the UK use to index those referents. However, there are also many terms with high log-frequency deviations that *are* linguistically distinctive. To isolate such terms, we began with the 400 terms with the highest estimated deviations, and then manually filtered this list, discarding Standard English words, proper nouns, numerals, and non-standard terms which had clear topical associations (e.g. ‘devo’: an abbreviation for ‘devolution’; ‘hh’: an acronym for ‘Hail Hail’, a football chant used by supporters of Celtic F.C.). The remaining 113 distinctively Scottish terms are listed in Table 2.

Almost three fourths of these terms are attested in the Scottish National Dictionary (SND) (Grant and Murison, 1931) or its online supplement (Scottish Language Dictionaries, 2004), which catalogue words that are distinctive to Scots (i.e. those which are not used, or are used differently, in Standard English), covering the period from the 1700s up to the present day. Many are also attested in the Dictionary of the Older Scottish Tongue (Aitken et al., 1990), which catalogues the entire vocabulary of Scots from the 1100s to the late 1600s. Of the attested Scots words, some are unique to Scots, e.g. BAIRNS (‘sons/daughters’), GREETIN (‘weeping’); some are cognates with English words that have fallen out of common usage, e.g. CRABBIT (‘crabbed’; ‘ill-tempered’), FEART (‘feared’; ‘frightened/timid’); some are cognates with English words but have a wider range of senses, e.g. HUNNERS is cognate with ‘hundreds’, but used more generally to mean ‘lots’ as in “love you hunners”, “there was hunners to do”; and many differ only in form from their English cognates, e.g. AFF (‘off’) and BAW (‘ball’).

Of the 29 terms that are not attested in SND, 9 are spelling variants or derived forms of attested

Neutral hashtags: #IndyRef (46,491) #ScotlandDecides (2552) #BBCIndyref (1591) #ScotDecides (934) #BigBigDebate (676) #ScottishIndependence (583) #IndyPlan (296) #ScottishReferendum (239) #IndyReasons (180) #IndependentScotland (26)

Yes hashtags: #VoteYes (8463) #YesScotland (1453) #YesBecause (1312) #The45 (908) #YouYesYet (827) #YesScot (670) #ActiveYes (508) #HopeOverFear (325) #Yes2014 (321) #VoteYesScotland (256) #GoForItScotland (153) #The45Plus (138) #YesFlash (114) #GenYes (92) #YesVote (76) #1Year2Yes (56) #VoteAye (53) #FreeScotland (52) #SaorAlba (45) #YesGenerations (39) #RIPBetterTogether (36) #NHSForYes (24) #AnotherScotlandIsPossible (23) #EndLondonRule (13)

No hashtags: #BetterTogether (2342) #NoThanks (1103) #VoteNo (867) #LabourNo (333) #LetsStayTogether (145) #VoteNo2014 (92) #UKOK (86) #VoteNoScotland (45) #JustSayNaw (43) #VoteNaw (42) #NoScotland (34) #DayOfUnity (30) #MaintainTheUnion (9)

Table 1: Hashtags related to the Scottish Independence Referendum and their frequencies in the IT dataset

Scots words, e.g. CANA, CANNY, and CANI are alternative spellings of the attested CANNAE, and WANTY is a contracted form of ‘want to’, analogous to the attested GONNAE and GONY. A further 5 are orthographic representations of distinctively Scottish pronunciations, e.g. ANO (‘I know’), HING (‘thing’); and 2 are acronyms for distinctively Scottish turns of phrase: GTF (‘Get Tae Fuck’) and MWI (‘Mad Wae It’). The final 13 could be described as contemporary Scottish slang, and include abbreviations: BEVY (‘beverage’)⁶, DEFOS (‘definitely’); drug-related lexis: WHITEY, ECCIES; profanities: BOABY, FANNYS; and everyday affective and descriptive words: DYN0 (‘amazing’), ROASTER (‘idiot’).

4.2 Lexical variables

Our goal is to measure the rate at which people index their Scottishness (either consciously or subconsciously) through the use of distinctively Scottish words, and to find out whether this rate varies across different groups of users (Yes hashtag users vs. No hashtag users), or across different contexts (tweets which contain referendum-related hashtags vs. tweets that don’t).

Were we to directly compare the frequencies of our Scottish terms across different sets of tweets, it would be difficult to untangle differences in the rate at which users are indexing the *referents* of those terms from differences in the rate at which they are indexing their Scottishness. For example, if people use the term MASEL (‘myself’) with a lower frequency in one context than in another, this could be because they are modulating their use of distinctively Scottish terms in response to the context, but it could also be because they are modulating the

⁶While ‘bevy’ is also used colloquially for ‘beverage’ in other parts of the UK, in Scotland it is more frequent and can additionally be used as a mass noun (“I had so much bevy I couldn’t even carry it”), and as a verb (“I’d bevy with him every weekend”).

rate at which they talk about themselves. To avoid this confound, we instead compare the *conditional* probabilities with which Scottish terms are used, given that their referents are being indexed at all.

We therefore consider only those Scottish terms for which we can identify semantically equivalent Standard English variants. We require that each variant of a given variable indexes the same set of senses and can occur in the same set of contexts, so for example we do not include YOUS as a variant of YOU, since while Scottish YI and Standard English YOU can index both the singular and plural second person pronouns, YOUS is only used for the plural. We also did not include variants of YES and NO since their use could be influenced by campaign slogans (e.g., the hashtags #VoteAye and #JustSayNaw). Our variables are listed in Table 3.

5 Study 1: Scotland-specific vocabulary usage on either side of the debate

Do tweeters who use Yes hashtags use Scottish variants at a higher rate than tweeters who use No hashtags, either when using these hashtags, or in general?

5.1 Method

We assign users in the IT dataset to two groups, **Yes** and **No**, based on the quantity $\frac{n_{u,yes}}{n_{u,yes}+n_{u,no}}$, where $n_{u,yes}$ is the number of tweets in which user u has used at least one of the Yes hashtags and none of the No hashtags in Table 1; and $n_{u,no}$ is the number of tweets in which u has used at least one No hashtag and none of the Yes hashtags. The **Yes** group consists of all users for whom this quantity is greater than or equal to 0.75, while the **No** group consists of all users for whom it is less than or equal to 0.25. Users for whom the value lies between 0.25 and 0.75 (as well as those for whom our dataset does not contain any tweets with Yes or No hashtags), are not assigned to either group. The **Yes** group

Acronyms: GTF MWI

Closed Class Words: ABOUT AE AFF ATS DAE FAE HAE MASEL MASELF OAN OOR OOT TAE WAE WAN WI WIS YERSEL YI YIN YOUS

Contractions CANNAE CANNI CANY CANA DEH DINI DINNY DIDNY DOESNY GONNAE GONY ISNY WANTY YER YIR

Discourse Markers: ACH ANAW ANO AWRIGHT AWRITE AWRYT AYE EH NAE NAW OOFT YASS YASSS YASSSS YASSSSS YIP

Open Class Words: AULD AWFY BAIRNS BAW BAWs BELTER BELTERS BEVY BOABY BOKE BRAW BURD BURDS CRABBIT DAFTY DAIN DEFOS DOON DUGS DYNO ECCIES FANNYS FEART FITBA FUD GAD GAWN GEES GID GRANDA GREETIN HAME HAW HING HINK HOOSE HOWLIN HUNNERS JIST LADDIE LASSIE LASSIES MANKY MAW MAWS MORRA MONGO PISH PISHED PISHING RAGIN ROASTER SARE SHITE SHITEY STEAMIN SUHIN WEANS WHITEY

Table 2: Scotland-specific vocabulary. Standard English equivalents of many words are shown in Table 3.

contains 4,513 users, while the *No* group contains 1,356 users, which is consistent with the general perception at the time that the Yes campaign was much more vocal than the No campaign. To test our hypothesis that the probability of choosing Scottish variants is, on average, greater for users in the *Yes* group than for users in the *No* group, we estimate the difference between the two groups in the average probability of choosing Scottish variants, and conduct a permutation test to approximate the distribution of this difference under the null hypothesis. We first test whether the *Yes* group are more likely than the *No* group to use Scottish variants in tweets which contain hashtags that indicate a stance on the referendum. Subsequently, we test whether the *Yes* group are more likely than the *No* group to use Scottish variants in general across all of their tweets.

5.1.1 Test statistic

Let U_g be the set of all users in group $g \in \{yes, no\}$ who have used at least one of the variables in Table 3. For a given user $u \in U_g$, let V be the set of all variables that u has used in at least one tweet. We estimate the probability of user u choosing a Scottish variant of variable $v \in V$ as $\hat{p}_{u,v} = \frac{n_{u,vscot}}{n_{u,v}}$, where $n_{u,vscot}$ is the token count of Scottish variants of v in user u 's tweets, and $n_{u,v}$ is the token count of all variants of v in user u 's tweets. Averaging across variables, we obtain $\hat{p}_u = \frac{1}{|V|} \sum_{v \in V} \hat{p}_{u,v}$. We then average across users to obtain the group mean, $\hat{p}_g = \frac{1}{|U_g|} \sum_{u \in U_g} \hat{p}_u$. Our test statistic is the difference between the two group means, $d = \hat{p}_{yes} - \hat{p}_{no}$.

5.1.2 Permutation test

We randomly shuffle users between the two groups (maintaining each group's original number of users), and re-compute the value of d using these permuted groups. We repeat this procedure 100,000 times in order to approximate the distri-

Group	Tweets w/ Yes or No hashtags		All tweets	
	Yes	No	Yes	No
# Users	3776	1121	4352	1322
# Tweets	10,436	2411	173,171	80,736

Table 4: Number of users and tweets included per group in the two analyses in Study 1

bution of differences in group means that would be observable were the difference independent of the assignment of users to groups. The proportion of permuted differences which are greater than or equal to the observed difference between the original group means provides an approximate p-value.

5.2 Results

For a tweet to be included in the analysis, it must contain at least one of the variables in Table 3. Hence not all users contribute data to the test statistic, as some have not used any of the variables in their tweets. The number of tweets and users included in each analysis are shown in Table 4.

The results for the first analysis are shown in the left column of Table 5. The difference between the two groups in their average probability of choosing Scottish variants in tweets that contain polarised referendum hashtags is statistically significant ($p < 0.002$). Results for the second analysis are shown in the right column of Table 5. Once again, the difference between the two groups is statistically significant ($p < 0.001$).

5.3 Discussion

The results show that the *Yes* group do use Scottish variants at a significantly higher rate than the *No* group, both when using Yes or No hashtags, and in general. The stronger significance level for the 'All tweets' dataset is partly due to its larger size (see Table 4), which enables better estimates of the

Variable	Scottish variants (freq. per million words)	Standard English variants (freq. per million words)
ABOUT	ABOOT (50)	ABOUT (2562)
ALRIGHT	AWRIGHT (10), AWRITE (17), AWRYT (17)	ALRIGHT (77), ALL RIGHT (4)
BALL	BAW (11)	BALL (116)
BALLS	BAWS (17)	BALLS (47)
BIRD	BURD (35)	BIRD (78)
BIRDS	BURDS (31)	BIRDS (44)
DEFINITELY	DEFOS (27)	DEFINITIELY (217)
DIDNT	DIDNY (26)	DIDNT (563), DID NOT (31)
DO	DAE (61)	DO (2712)
DOESNT	DOESNY (18)	DOESNT (433), DOES NOT (33)
DOGS	DUGS (11)	DOGS (69)
DOING	DAIN (17)	DOING (590)
DONT	DEH (12), DINI (12), DINNY (62)	DONT (2880), DO NOT (92)
DOWN	DOON (49)	DOWN (786)
FOOTBALL	FITBA (13)	FOOTBALL (289)
FROM	FAE (77)	FROM (2485)
GIVES	GEES (14)	GIMME (5), GIVE ME (108), GIVE US (21), GIVES (75)
GOING	GAWN (15)	GOING (1884)
GOOD	GID (82)	GOOD (2602)
GRANDAD	GRANDA (7)	GRANDAD (19), GRANDFATHER (5), GRANDPA (9)
HAVE	HAE (9)	HAVE (4549)
HOME	HAME (22)	HOME (832)
HOUSE	HOOSE (20)	HOUSE (463)
I KNOW	ANO (42)	I KNOW (556)
ISNT	ISNY (16)	ISNT (342), IS NOT (151)
JUST	JIST (7)	JUST (5550)
MYSELF	MASEL (14), MASELF (15)	MYSELF (553)
OF	AE (75)	OF (9186)
OFF	AFF (82)	OFF (1567)
OLD	AULD (28)	OLD (526)
ON	OAN (38)	ON (7782)
ONE	WAN (33), YIN (28)	ONE(2537)
OUR	OOR (14)	OUR (790)
OUT	OOT (181)	OUT (3053)
PISSED	PISHED (19)	PISSED (66)
PISSING	PISHING (12)	PISSING (32)
SHIT	SHITE (428)	SHIT (764)
SHITY	SHITEY (25)	SHITY (52)
SOMETHING	SUHIN (17)	SOMETHING (614)
SORE	SARE (13)	SORE (140)
THATS	ATS (9)	THATS (1405)
THING	HING (11)	THING (749)
THINK	HINK (34)	THINK (1939)
TO	TAE (186)	TO (19996), TOO (1629)
TOMORROW	MORRA (27)	TOMORROW (1183)
WANT TO	WANTY (52)	WANNA (284), WANT TO (940)
WAS	WIS (33)	WAS (4197)
WITH	WI (85), WAE (116)	WITH (4774)
YOU	YI (26)	YOU (10891)
YOUR	YER (237), YIR (11)	YOUR (3094), YOURE (915), YOU ARE (342)
YOURSELF	YERSEL (11)	YOURSELF (193)

Table 3: Variables used in our studies, with each variant’s frequency per million tokens in the GS dataset

	Tweets w/ Yes or No hashtags	All tweets
\hat{p}_{yes}	0.00766	0.01443
\hat{p}_{no}	0.00211	0.00734
d	0.00555	0.00709
p -value	0.00103	0.00001

Table 5: Results of the two analyses in Study 1

usage rates. While the rates are very low overall, the relative differences are large: the **Yes** group rate is more than three times the **No** group rate when we include only tweets with Yes or No hashtags, and approximately twice as big when we include all tweets. The higher rates in the ‘All Tweets’ dataset suggest that both groups of users chose Scottish variants less often when discussing the referendum than in their other tweets. However, the test we used does not provide a significance value for the difference in usage rates across the two datasets. To establish whether users do modulate their usage of Scottish variants when discussing the referendum, we will need a more careful paired design.

6 Study 2: Effects of topic and audience on Scotland-specific vocabulary usage

Do tweeters choose Scottish variants at a different rate when using referendum-related hashtags than in their other tweets?

6.1 Method

We need a statistic that corrects for the fact that some variables might have higher rates of Scottish variants than others. For example if users tend to produce Scottish variants of variable v_1 at a higher rate than for v_2 , and use v_1 more in tweets that don’t contain referendum-related hashtags, then it could appear that users are suppressing their Scottish usage in referendum-related tweets when in fact this is a lexical effect.

Let U be the set of all users who have used at least one of the variables in Table 3 in both a tweet that contains a referendum-related hashtag (i.e. a tweet that belongs to the IT dataset, referred to hereafter as an Indyref tweet) and in a tweet that does not contain a referendum-related hashtag (referred to hereafter as a Control tweet). For a given user $u \in U$, let V be the set of all variables that u has used in at least one Indyref tweet, and in at least one Control tweet. Let $\hat{p}_{I,v}$ for user u be the

estimated probability that u chooses a Scottish variant of variable $v \in V$, conditioned on the fact that she is using variable v in an Indyref tweet. Analogously, let $\hat{p}_{C,v}$ be the estimated probability that u chooses a Scottish variant of variable v , conditioned on the fact that she is using variable v in a Control tweet. The difference in user u ’s probability of choosing a Scottish variant of variable v in an Indyref tweet and in a Control tweet is then $d_v = \hat{p}_{I,v} - \hat{p}_{C,v}$. Averaging across all variables, we define $d_u = \frac{1}{|V|} \sum_{v \in V} d_v$.

The null hypothesis is that on average, users are no more or less likely to choose Scottish variants in Indyref tweets than in Control tweets. Therefore, under the null hypothesis, the mean value of d_u across all users, $\bar{d}_u = \frac{1}{|U|} \sum_{u \in U} d_u$, would be zero. We perform a one-sample t-test to determine whether \bar{d}_u is significantly different than zero.

We use this method to conduct two separate analyses. In the first analysis, our pool of Control tweets is the set of *all* tweets from the original filtered dataset that do not contain any of the hashtags in Table 1. In the second analysis, we limit our pool of Control tweets to those which do not contain any of the hashtags from Table 1, but *do* contain at least one other hashtag. This second analysis is designed to test whether the recent finding that US Twitter users are less likely to use regionally-specific words in tweets which contain hashtags (Pavalanathan and Eisenstein, 2015) applies to Scottish users as well.

6.2 Results

The number of tweets and users that were included in each analysis are shown in Table 6.

Results for the first analysis are shown in the left column of Table 7. The difference is statistically significant ($p < 0.01$), indicating that on average, individuals are less likely to choose Scottish variants when using referendum-related hashtags than in their other tweets. Results for the second analysis are shown in the right column of Table 7. In this case, the difference is not statistically significant.

6.3 Discussion

In light of (a) the apparent relationship between national identity and constitutional preference, (b) the history of Scots as the prestige language of a previously-independent Scotland, supplanted by English in large part due to the birth of the United Kingdom, and (c) the results of Study 1, which indicate that pro-independence users choose Scottish variants at a significantly higher rate than anti-

	All Controls	Controls w/ Hashtags
# Users	11,011	7429
# Indyref Tweets	41,924	35,241
# Control Tweets	693,815	195,145

Table 6: Number of users and tweets included in the two analyses in Study 2

	All Controls	Controls w/ Hashtags
\bar{d}_u	-0.0015	-0.0010
std error	0.0005	0.0006
<i>t</i> -statistic	-2.996	-1.758
<i>p</i> -value	0.0027	0.0788

Table 7: Results of the two analyses in Study 2

independence users—it may at first appear surprising that people are *less* likely to choose Scottish variants in tweets containing referendum-related hashtags than in their other tweets.

It is conceivable that *Yes* users increase their rate of Scottish variants in Indyref tweets whilst *No* users decrease it, such that their effects cancel out; but since *Yes* users are more prolific in the IT dataset, if anything we would expect this imbalance to make the effect even more positive. The fact that we see a significant negative effect in spite of the greater number of *Yes* tweets means we can be reasonably confident that even if *Yes* users aren't significantly reducing their usage of Scottish variants in Indyref tweets, they certainly aren't increasing it.

It is also worth noting that we did not exhaustively identify every hashtag that has been used in relation to the referendum, so inevitably there will be some tweets with referendum-related hashtags in the Control set (such as example tweet (3) in §2), and there may also be some non-referendum tweets in the Indyref set. However, if anything this would dilute any differences between the two lists, yet we still find an effect.

The fact that this effect does not reach significance when we remove Control tweets without hashtags suggests that the primary reason users are reducing their rate of Scottish variants in Indyref tweets is not because of the *topic* under discussion, but because the use of hashtags broadens the potential audience. This explanation accords with Pavalanathan and Eisenstein's (2015) finding that

amongst Twitter users in the US, non-standard and regional variants are less likely to be used in tweets that target larger audiences. Of course, it is possible that topic has an effect as well, but the present study does not provide evidence for that conclusion.

7 Conclusion

We presented the first large-scale study of distinctively Scottish language use on social media, showing that this use includes a mixture of traditional Scots vocabulary, newer Scottish slang, and alternative spellings that reflect Scottish pronunciation. We also studied how users' language might reflect their political views and discourse. We showed that *Yes* users use Scottish variants at a higher rate than *No* users, whether discussing the independence referendum or not. But overall, users tend to decrease their use of Scottish variants when discussing the referendum. This result suggests that although *Yes* users generally express a stronger Scottish linguistic identity than *No* users, they are not choosing to express this identity strongly in political discourse aimed at a broad audience. Due to the very low rates of Scottish variants overall, our data set is too small to study differences between individual variables or even conclusively say whether there may be effects of both topic and audience size on the use of Scottish language. However, we hope to be able to answer these questions in future by collecting a more complete set of data for the particular users studied here.

8 Acknowledgements

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

- Adam J. Aitken, James A.C. Stevenson, Sir William Alexander Craigie, and Margaret G. Dareau. 1990. *A Dictionary of the Older Scottish Tongue Vols. 1-7: From the Twelfth Century to the End of the Seventeenth*. MacMillan Publishing Company.
- David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.
- Lin Su Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social

- media: A case study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130. Association for Computational Linguistics.
- John Corbett, J. Derrick McClure, and Jane Stuart-Smith. 2003. A brief history of Scots. In John Corbett, J. Derrick McClure, and Jane Stuart-Smith, editors, *The Edinburgh Companion to Scots*, pages 1–16. Edinburgh, Edinburgh University Press.
- Gabriel Doyle. 2014. Mapping dialectal variation by querying social media. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 98–106. Association for Computational Linguistics.
- Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the International Conference on Machine Learning*, pages 1041–1048.
- Jacob Eisenstein. 2015. Systematic patterning in phonologically-motivated orthographic variation. *Journal of Sociolinguistics*, 19(2):161–188.
- Bruno Gonçalves and David Sánchez. 2014. Crowdsourcing dialect characterization through Twitter. *PloS one*, 9(11):e112074.
- William Grant and David D. Murison. 1931. *The Scottish National Dictionary*. Scottish National Dictionary Association.
- William Grant. 1931. Phonetic description of Scottish language and dialects. In *The Scottish National Dictionary*, volume 1, pages 9–41. Online: <http://www.dsl.ac.uk/about-scots/history-of-scots/>.
- Yuan Huang, Diansheng Guo, Alice Kasakoff, and Jack Grieve. 2015. Understanding US regional linguistic variation with Twitter data analysis. *Computers, environment and urban systems*.
- Taylor Jones. 2015. Toward a description of African American Vernacular English dialect regions using “Black Twitter”. *American Speech*, 90(4):403–440.
- Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of studying and processing dialects in social media. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 9–18. Association for Computational Linguistics.
- Billy Kay. 1988. *Scots: The Mither Tongue*. Grafton, first edition.
- Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.
- Caroline McAfee. 1985. Nationalism and the Scots renaissance now. In Manfred Görlach, editor, *Focus on: Scotland (Varieties of English around the world, V.5)*, pages 7–16. Amsterdam/Philadelphia, John Benjamins Publishing Company.
- Dong-Phuong Nguyen, RB Trieschnigg, and Leonie Cornips. 2015. Audience and the use of minority languages on Twitter. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media*, pages 666–669.
- Umashanthi Pavalanathan and Jacob Eisenstein. 2015. Audience-modulated variation in online social media. *American Speech*, 90(2):187–213.
- ScotCen. 2013. Should Scotland be an independent country? (combined responses of those who have and those who haven’t decided yet) broken down by ‘Moreno’ national identity. Retrieved from: <http://whatscotlandthinks.org/>. Accessed: 2016-09-30.
- Scottish Language Dictionaries. 2004. Dictionary of the Scots language. <http://www.dsl.ac.uk/>. Accessed: 2016-12-20.
- Rachael Tatman. 2015. #go awn: Sociophonetic variation in variant spellings on Twitter. *Working Papers of the Linguistics Circle of the University of Victoria*, 25(2):97–108.

What Do Recurrent Neural Network Grammars Learn About Syntax?

Adhiguna Kuncoro[♠] Miguel Ballesteros[◇] Lingpeng Kong[♠]

Chris Dyer^{♠♣} Graham Neubig[♠] Noah A. Smith[♡]

[♠]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

[◇]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

[♣]DeepMind, London, UK

[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA

{akuncoro, lingpenk, gneubig}@cs.cmu.edu

miguel.ballesteros@ibm.com, cdyer@google.com, nasmith@cs.washington.edu

Abstract

Recurrent neural network grammars (RNNG) are a recently proposed probabilistic generative modeling family for natural language. They show state-of-the-art language modeling and parsing performance. We investigate what information they learn, from a linguistic perspective, through various ablations to the model and the data, and by augmenting the model with an attention mechanism (GA-RNNG) to enable closer inspection. We find that explicit modeling of composition is crucial for achieving the best performance. Through the attention mechanism, we find that headedness plays a central role in phrasal representation (with the model’s latent attention largely agreeing with predictions made by hand-crafted head rules, albeit with some important differences). By training grammars without nonterminal labels, we find that phrasal representations depend minimally on nonterminals, providing support for the endocentricity hypothesis.

1 Introduction

In this paper, we focus on a recently proposed class of probability distributions, recurrent neural network grammars (RNNGs; Dyer et al., 2016), designed to model syntactic derivations of sentences. We focus on RNNGs as generative probabilistic models over trees, as summarized in §2.

Fitting a probabilistic model to data has often been understood as a way to test or confirm some aspect of a theory. We talk about a model’s assumptions and sometimes explore its parameters or posteriors over its latent variables in order to gain understanding of what it “discovers” from the

data. In some sense, such models can be thought of as mini-scientists.

Neural networks, including RNNGs, are capable of representing larger classes of hypotheses than traditional probabilistic models, giving them more freedom to explore. Unfortunately, they tend to be bad mini-scientists, because their parameters are difficult for human scientists to interpret.

RNNGs are striking because they obtain state-of-the-art parsing and language modeling performance. Their relative lack of independence assumptions, while still incorporating a degree of linguistically-motivated prior knowledge, affords the model considerable freedom to derive its own insights about syntax. If they are mini-scientists, the discoveries they make should be of particular interest as propositions about syntax (at least for the particular genre and dialect of the data).

This paper manipulates the inductive bias of RNNGs to test linguistic hypotheses.¹ We begin with an ablation study to discover the importance of the composition function in §3. Based on the findings, we augment the RNNG composition function with a novel **gated attention** mechanism (leading to the GA-RNNG) to incorporate more interpretability into the model in §4. Using the GA-RNNG, we proceed by investigating the role that individual heads play in phrasal representation (§5) and the role that nonterminal category labels play (§6). Our key findings are that lexical heads play an important role in representing most phrase types (although compositions of multiple salient heads are not infrequent, especially

¹RNNGs have less inductive bias relative to traditional unlexicalized probabilistic context-free grammars, but more than models that parse by transducing word sequences to linearized parse trees represented as strings (Vinyals et al., 2015). Inductive bias is necessary for learning (Mitchell, 1980); we believe the important question is not “how little can a model get away with?” but rather the benefit of different forms of inductive bias as data vary.

for conjunctions) and that nonterminal labels provide little additional information. As a by-product of our investigation, a variant of the RNNG without ensembling achieved the best reported supervised phrase-structure parsing (93.6 F_1 ; English PTB) and, through conversion, dependency parsing (95.8 UAS, 94.6 LAS; PTB SD). The code and pretrained models to replicate our results are publicly available².

2 Recurrent Neural Network Grammars

An RNNG defines a *joint* probability distribution over string terminals and phrase-structure nonterminals.³ Formally, the RNNG is defined by a triple $\langle N, \Sigma, \Theta \rangle$, where N denotes the set of nonterminal symbols (NP, VP, etc.), Σ the set of all terminal symbols (we assume that $N \cap \Sigma = \emptyset$), and Θ the set of all model parameters. Unlike previous works that rely on hand-crafted rules to compose more fine-grained phrase representations (Collins, 1997; Klein and Manning, 2003), the RNNG implicitly parameterizes the information passed through compositions of phrases (in Θ and the neural network architecture), hence weakening the strong independence assumptions in classical probabilistic context-free grammars.

The RNNG is based on an abstract state machine like those used in transition-based parsing, with its algorithmic state consisting of a stack of partially completed constituents, a buffer of already-generated terminal symbols, and a list of past actions. To generate a sentence x and its phrase-structure tree y , the RNNG samples a sequence of actions to construct y top-down. Given y , there is one such sequence (easily identified), which we call the oracle, $\mathbf{a} = \langle a_1, \dots, a_n \rangle$ used during supervised training.

The RNNG uses three different actions:

- $\text{NT}(X)$, where $X \in N$, introduces an open nonterminal symbol onto the stack, e.g., “(NP”;
- $\text{GEN}(x)$, where $x \in \Sigma$, generates a terminal symbol and places it on the stack and buffer; and
- REDUCE indicates a constituent is now complete. The elements of the stack that comprise the current constituent (going back to the last

²<https://github.com/clab/rnng/tree/master/interpreting-rnng>

³Dyer et al. (2016) also defined a conditional version of the RNNG that can be used only for parsing; here we focus on the generative version since it is more flexible and (rather surprisingly) even learns better estimates of $p(\mathbf{y} | \mathbf{x})$.

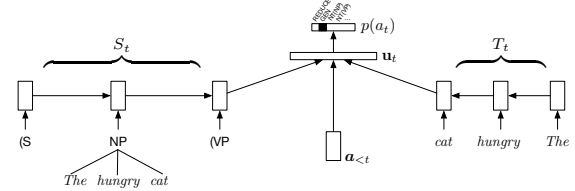


Figure 1: The RNNG consists of a stack, buffer of generated words, and list of past actions that lead to the current configuration. Each component is embedded with LSTMs, and the parser state summary \mathbf{u}_t is used as top-layer features to predict a softmax over all feasible actions. This figure is due to Dyer et al. (2016).

open nonterminal) are popped, a **composition function** is executed, yielding a composed representation that is pushed onto the stack.

At each timestep, the model encodes the stack, buffer, and past actions, with a separate LSTM (Hochreiter and Schmidhuber, 1997) for each component as features to define a distribution over the next action to take (conditioned on the full algorithmic state). The overall architecture is illustrated in Figure 1; examples of full action sequences can be found in Dyer et al. (2016).

A key element of the RNNG is the composition function, which reduces a completed constituent into a single element on the stack. This function computes a vector representation of the new constituent; it also uses an LSTM (here a bidirectional one). This composition function, which we consider in greater depth in §3, is illustrated in Fig. 2.

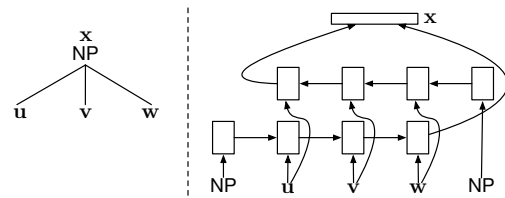


Figure 2: RNNG composition function on each REDUCE operation; the network on the right models the structure on the left (Dyer et al., 2016).

Since the RNNG is a generative model, it attempts to maximize $p(\mathbf{x}, \mathbf{y})$, the *joint* distribution

of strings and trees, defined as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{a}) = \prod_{t=1}^n p(a_t \mid a_1, \dots, a_{t-1}).$$

In other words, $p(\mathbf{x}, \mathbf{y})$ is defined as a product of local probabilities, conditioned on all past actions. The joint probability estimate $p(\mathbf{x}, \mathbf{y})$ can be used for both phrase-structure parsing (finding $\arg \max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$) and language modeling (finding $p(\mathbf{x})$ by marginalizing over the set of possible parses for \mathbf{x}). Both inference problems can be solved using an importance sampling procedure.⁴ We report all RNNG performance based on the corrigendum to Dyer et al. (2016).

3 Composition is Key

Given the same data, under both the discriminative and generative settings RNNGs were found to parse with significantly higher accuracy than (respectively) the models of Vinyals et al. (2015) and Choe and Charniak (2016) that represent \mathbf{y} as a “linearized” sequence of symbols and parentheses without explicitly capturing the tree structure, or even constraining the \mathbf{y} to be a well-formed tree (see Table 1). Vinyals et al. (2015) directly predict the sequence of nonterminals, “shifts” (which consume a terminal symbol), and parentheses from left to right, conditional on the input terminal sequence \mathbf{x} , while Choe and Charniak (2016) used a sequential LSTM language model on the same linearized trees to create a generative variant of the Vinyals et al. (2015) model. The generative model is used to re-rank parse candidates.

Model	F_1
Vinyals et al. (2015) – PTB only	88.3
Discriminative RNNG	91.2
Choe and Charniak (2016) – PTB only	92.6
Generative RNNG	93.3

Table 1: Phrase-structure parsing performance on PTB §23. All results are reported using single-model performance and without any additional data.

The results in Table 1 suggest that the RNNG’s explicit composition function (Fig. 2), which

⁴Importance sampling works by using a proposal distribution $q(\mathbf{y} \mid \mathbf{x})$ that is easy to sample from. In Dyer et al. (2016) and this paper, the proposal distribution is the discriminative variant of the RNNG; see Dyer et al. (2016).

Vinyals et al. (2015) and Choe and Charniak (2016) must learn implicitly, plays a crucial role in the RNNG’s generalization success. Beyond this, Choe and Charniak’s generative variant of Vinyals et al. (2015) is another instance where generative models trained on the PTB outperform discriminative models.

3.1 Ablated RNNGs

On close inspection, it is clear that the RNNG’s three data structures—stack, buffer, and action history—are redundant. For example, the action history and buffer contents completely determine the structure of the stack at every timestep. Every generated word goes onto the stack, too; and some past words will be composed into larger structures, but through the composition function, they are all still “available” to the network that predicts the next action. Similarly, the past actions are redundant with the stack. Despite this redundancy, only the stack incorporates the composition function. Since each of the ablated models is sufficient to encode all necessary partial tree information, the primary difference is that ablations with the stack use explicit composition, to which we can therefore attribute most of the performance difference.

We conjecture that the stack—the component that makes use of the composition function—is critical to the RNNG’s performance, and that the buffer and action history are not. In transition-based parsers built on expert-crafted features, the most recent words and actions are useful if they are salient, although neural representation learners can automatically learn what information should be salient.

To test this conjecture, we train **ablated RNNGs** that lack each of the three data structures (action history, buffer, stack), as well as one that lacks both the action history and buffer.⁵ If our conjecture is correct, performance should degrade most without the stack, and the stack alone should perform competitively.

Experimental settings. We perform our experiments on the English PTB corpus, with §02–21 for training, §24 for validation, and §23 for test; no additional data were used for training. We fol-

⁵Note that the ablated RNNG without a stack is quite similar to Vinyals et al. (2015), who encoded a (partial) phrase-structure tree as a sequence of open and close parentheses, terminals, and nonterminal symbols; our action history is quite close to this, with each NT(X) capturing a left parenthesis and X nonterminal, and each REDUCE capturing a right parenthesis.

low the same hyperparameters as the generative model proposed in Dyer et al. (2016).⁶ The generative model did not use any pretrained word embeddings or POS tags; a discriminative variant of the standard RNNG was used to obtain tree samples for the generative model. All further experiments use the same settings and hyperparameters unless otherwise noted.

Experimental results. We trained each ablation from scratch, and compared these models on three tasks: English phrase-structure parsing (labeled F_1), Table 2; dependency parsing, Table 3, by converting parse output to Stanford dependencies (De Marneffe et al., 2006) using the tool by Kong and Smith (2014); and language modeling, Table 4. The last row of each table reports the performance of a novel variant of the (stack-only) RNNG with attention, to be presented in §4.

Model	F_1
Vinyals et al. (2015) [†]	92.1
Choe and Charniak (2016)	92.6
Choe and Charniak (2016) [†]	93.8
Baseline RNNG	93.3
<hr/>	
Ablated RNNG (no history)	93.2
Ablated RNNG (no buffer)	93.3
Ablated RNNG (no stack)	92.5
Stack-only RNNG	93.6
<hr/>	
GA-RNNG	93.5

Table 2: Phrase-structure parsing performance on PTB §23. [†] indicates systems that use additional unparsed data (semisupervised). The GA-RNNG results will be discussed in §4.

Discussion. The RNNG with only a stack is the strongest of the ablations, and it even outperforms the “full” RNNG with all three data structures. Ablating the stack gives the worst among the new results. This strongly supports the importance of the composition function: a proper REDUCE operation that transforms a constituent’s parts and non-terminal label into a single explicit (vector) representation is helpful to performance.

It is noteworthy that the stack alone is stronger than the original RNNG, which—in principle—can learn to disregard the buffer and action his-

⁶The model is trained using stochastic gradient descent, with a learning rate of 0.1 and a per-epoch decay of 0.08. All experiments with the generative RNNG used 100 tree samples for each sentence, obtained by sampling from the local softmax distribution of the discriminative RNNG.

Model	UAS	LAS
Kiperwasser and Goldberg (2016)	93.9	91.9
Andor et al. (2016)	94.6	92.8
Dozat and Manning (2016)	95.4	93.8
Choe and Charniak (2016) [†]	95.9	94.1
Baseline RNNG	95.6	94.4
<hr/>		
Ablated RNNG (no history)	95.4	94.2
Ablated RNNG (no buffer)	95.6	94.4
Ablated RNNG (no stack)	95.1	93.8
Stack-only RNNG	95.8	94.6
<hr/>		
GA-RNNG	95.7	94.5

Table 3: Dependency parsing performance on PTB §23 with Stanford Dependencies (De Marneffe and Manning, 2008). [†] indicates systems that use additional unparsed data (semisupervised).

tory. Since the stack maintains syntactically “recent” information near its top, we conjecture that the learner is overfitting to spurious predictors in the buffer and action history that explain the training data but do not generalize well.

A similar performance degradation is seen in language modeling (Table 4): the stack-only RNNG achieves the best performance, and ablating the stack is most harmful. Indeed, modeling syntax without explicit composition (the stack-ablated RNNG) provides little benefit over a sequential LSTM language model.

Model	Test ppl. (PTB)
IKN 5-gram	169.3
LSTM LM	113.4
RNNG	105.2
<hr/>	
Ablated RNNG (no history)	105.7
Ablated RNNG (no buffer)	106.1
Ablated RNNG (no stack)	113.1
Stack-only RNNG	101.2
<hr/>	
GA-RNNG	100.9

Table 4: Language modeling: perplexity. IKN refers to Kneser-Ney 5-gram LM.

We remark that the stack-only results are the best published PTB results for both phrase-structure and dependency parsing among supervised models.

4 Gated Attention RNNG

Having established that the composition function is key to RNNG performance (§3), we now seek to understand the nature of the composed phrasal representations that are learned. Like most neural networks, interpreting the composition function’s

behavior is challenging. Fortunately, linguistic theories offer a number of hypotheses about the nature of representations of phrases that can provide a conceptual scaffolding to understand them.

4.1 Linguistic Hypotheses

We consider two theories about phrasal representation. The first is that phrasal representations are strongly determined by a privileged lexical head. Augmenting grammars with lexical head information has a long history in parsing, starting with the models of Collins (1997), and theories of syntax such as the “bare phrase structure” hypothesis of the Minimalist Program (Chomsky, 1993) posit that phrases are represented purely by single lexical heads. Proposals for multiple headed phrases (to deal with tricky cases like conjunction) likewise exist (Jackendoff, 1977; Keenan, 1987). Do the phrasal representations learned by RNNGs depend on individual lexical heads or multiple heads? Or do the representations combine all children without any salient head?

Related to the question about the role of heads in phrasal representation is the question of whether phrase-internal material wholly determines the representation of a phrase (an endocentric representation) or whether nonterminal relabeling of a constituent introduces new information (exocentric representations). To illustrate the contrast, an endocentric representation is representing a noun phrase with a noun category, whereas $S \rightarrow NP VP$ exocentrically introduces a new syntactic category that is neither NP nor VP (Chomsky, 1970).

4.2 Gated Attention Composition

To investigate what the stack-only RNNG learns about headedness (and later endocentricity), we propose a variant of the composition function that makes use of an explicit attention mechanism (Bahdanau et al., 2015) and a sigmoid gate with multiplicative interactions, henceforth called **GA-RNNG**.

At every REDUCE operation, the GA-RNNG assigns an “attention weight” to each of its children (between 0 and 1 such that the total weight off all children sums to 1), and the parent phrase is represented by the combination of a sum of each child’s representation scaled by its attention weight and its nonterminal type. Our weighted sum is more expressive than traditional head rules, however, because it allows attention to be divided among multiple constituents. Head rules, con-

versely, are analogous to giving all attention to one constituent, the one containing the lexical head.

We now formally define the GA-RNNG’s composition function. Recall that \mathbf{u}_t is the concatenation of the vector representations of the RNNG’s data structures, used to assign probabilities to each of the actions available at timestep t (see Fig. 1, the layer before the softmax at the top). For simplicity, we drop the timestep index here. Let \mathbf{o}_{nt} denote the vector embedding (learned) of the nonterminal being constructed, for the purpose of computing attention weights.

Now let $\mathbf{c}_1, \mathbf{c}_2, \dots$ denote the sequence of vector embeddings for the constituents of the new phrase. The length of these vectors is defined by the dimensionality of the bidirectional LSTM used in the original composition function (Fig. 2). We use semicolon (;) to denote vector concatenation operations.

The attention vector is given by:

$$\mathbf{a} = \text{softmax} \left([\mathbf{c}_1 \ \mathbf{c}_2 \ \dots]^\top \mathbf{V} [\mathbf{u}; \mathbf{o}_{nt}] \right) \quad (1)$$

Note that the length of \mathbf{a} is the same as the number of constituents, and that this vector sums to one due to the softmax. It divides a single unit of attention among the constituents.

Next, note that the *constituent source vector* $\mathbf{m} = [\mathbf{c}_1; \mathbf{c}_2; \dots] \mathbf{a}$ is a convex combination of the child-constituents, weighted by attention. We will combine this with another embedding of the nonterminal denoted as \mathbf{t}_{nt} (separate from \mathbf{o}_{nt}) using a sigmoid gating mechanism:

$$\mathbf{g} = \sigma (\mathbf{W}_1 \mathbf{t}_{nt} + \mathbf{W}_2 \mathbf{m} + \mathbf{b}) \quad (2)$$

Note that the value of the gate is bounded between $[0, 1]$ in each dimension.

The new phrase’s final representation uses element-wise multiplication (\odot) with respect to both \mathbf{t}_{nt} and \mathbf{m} , a process reminiscent of the LSTM “forget” gate:

$$\mathbf{c} = \mathbf{g} \odot \mathbf{t}_{nt} + (1 - \mathbf{g}) \odot \mathbf{m}. \quad (3)$$

The intuition is that the composed representation should incorporate both nonterminal information and information about the constituents (through weighted sum and attention mechanism). The gate \mathbf{g} modulates the interaction between them to account for varying importance between the two in different contexts.

Experimental results. We include this model’s performance in Tables 2–4 (last row in all tables). It is clear that the model outperforms the baseline RNNG with all three structures present and achieves competitive performance with the strongest, stack-only, RNNG variant.

5 Headedness in Phrases

We now exploit the attention mechanism to probe what the RNNG learns about headedness on the WSJ §23 test set (unseen before by the model).

5.1 The Heads that GA-RNNG Learns

The attention weight vectors tell us which constituents are most important to a phrase’s vector representation in the stack. Here, we inspect the attention vectors to investigate whether the model learns to center its attention around a single, or by extension a few, salient elements, which would confirm the presence of headedness in GA-RNNG.

First, we consider several major nonterminal categories, and estimate the average *perplexity* of the attention vectors. The average perplexity can be interpreted as the average number of “choices” for each nonterminal category; this value is only computed for the cases where the number of components in the composed phrase is at least two (otherwise the attention weight would be trivially 1). The minimum possible value for the perplexity is 1, indicating a full attention weight around one component and zero everywhere else.

Figure 3 (in blue) shows much less than 2 average “choices” across nonterminal categories, which also holds true for all other categories not shown. For comparison we also report the average perplexity of the uniform distribution for the same nonterminal categories (Fig. 3 in red); this represents the highest entropy cases where there is no headedness at all by assigning the same attention weight to each constituent (e.g. attention weights of 0.25 each for phrases with four constituents). It is clear that the learned attention weights have much lower perplexity than the uniform distribution baseline, indicating that the learned attention weights are quite peaked around certain components. This implies that phrases’ vectors tend to resemble the vector of one salient constituent, but not exclusively, as the perplexity for most categories is still not close to one.

Next, we consider the how attention is distributed for the major nonterminal categories in

Table 5, where the first five rows of each category represent compositions with highest entropy, and the next five rows are qualitatively analyzed. The high-entropy cases where the attention is most divided represent more complex phrases with conjunctions or more than one plausible head.

NPs. In most simple noun phrases (representative samples in rows 6–7 of Table 5), the model pays the most attention to the *rightmost noun* and assigns near-zero attention on determiners and possessive determiners, while also paying nontrivial attention weights to the adjectives. This finding matches existing head rules and our intuition that nouns head noun phrases, and that adjectives are more important than determiners.

We analyze the case where the noun phrase contains a conjunction in the last three rows of Table 5. The syntax of conjunction is a long-standing source of controversy in syntactic analysis (Johannessen, 1998, *inter alia*). Our model suggests that several representational strategies are used, when coordinating single nouns, both the first noun (8) and the last noun (9) may be selected. However, in the case of conjunctions of multiple noun *phrases* (as opposed to multiple single-word nouns), the model consistently picks the conjunction as the head. All of these representational strategies have been argued for individually on linguistic grounds, and since we see all of them present, RNNGs face the same confusion that linguists do.

VPs. The attention weights on simple verb phrases (e.g., “VP → V NP”, 9) are peaked around the noun phrase instead of the verb. This implies that the verb phrase would look most similar to the noun under it and contradicts existing head rules that unanimously put the verb as the head of the verb phrase. Another interesting finding is that the model pays attention to *polarity* information, where negations are almost always assigned nontrivial attention weights.⁷ Furthermore, we find that the model attends to the conjunction terminal in conjunctions of verb phrases (e.g., “VP → VP and VP”, 10), reinforcing the similar finding for conjunction of noun phrases.

PPs. In almost all cases, the model attends to the preposition terminal instead of the noun phrases or complete clauses under it, regardless of the type of preposition. Even when the preposi-

⁷Cf. Li et al. (2016), where sequential LSTMs discover polarity information in sentiment analysis, although perhaps more surprising as polarity information is less intuitively central to syntax and language modeling.

tional phrase is only used to make a connection between two noun phrases (e.g., “PP → NP after NP”, 10), the prepositional connector is still considered the most salient element. This is less consistent with the Collins and Stanford head rules, where prepositions are assigned a lower priority when composing PPs, although more consistent with the Johansson head rule (Johansson and Nugues, 2007).

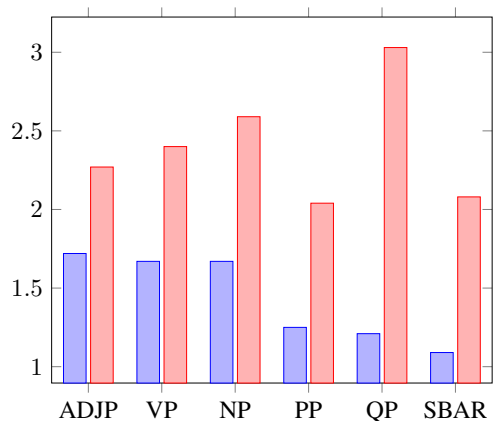


Figure 3: Average perplexity of the learned attention vectors on the test set (blue), as opposed to the average perplexity of the uniform distribution (red), computed for each major phrase type.

5.2 Comparison to Existing Head Rules

To better measure the overlap between the attention vectors and existing head rules, we converted the trees in PTB §23 into a dependency representation using the attention weights. In this case, the attention weight functions as a “dynamic” head rule, where all other constituents within the same composed phrase are considered to modify the constituent with the highest attention weight, repeated recursively. The head of the composed representation for “S” at the top of the tree is attached to a special root symbol and functions as the head of the sentence.

We measure the overlap between the resulting tree and conversion results of the same trees using the Collins (1997) and Stanford dependencies (De Marneffe et al., 2006) head rules. Results are evaluated using the standard evaluation script (excluding punctuation) in terms of UAS, since the attention weights do not provide labels.

Results. The model has a higher overlap with the conversion using Collins head rules (49.8

UAS) rather than the Stanford head rules (40.4 UAS). We attribute this large gap to the fact that the Stanford head rules incorporate more semantic considerations, while the RNNG is a purely syntactic model. In general, the attention-based tree output has a high error rate ($\approx 90\%$) when the dependent is a verb, since the constituent with the highest attention weight in a verb phrase is often the noun phrase instead of the verb, as discussed above. The conversion accuracy is better for nouns ($\approx 50\%$ error), and much better for determiners (30%) and particles (6%) with respect to the Collins head rules.

Discussion. GA-RNNG has the power to infer head rules, and to a large extent, it does. It follows some conventions that are established in one or more previous head rule sets (e.g., prepositions head prepositional phrases, nouns head noun phrases), but attends more to a verb phrase’s nominal constituents than the verb. It is important to note that this is not the by-product of learning a specific model for parsing; the training objective is *joint* likelihood, which is not a proxy loss for parsing performance. These decisions were selected because they make the data maximally likely (though admittedly only locally maximally likely). We leave deeper consideration of this noun-centered verb phrase hypothesis to future work.

6 The Role of Nonterminal Labels

Emboldened by our finding that GA-RNNGs learn a notion of headedness, we next explore whether heads are sufficient to create representations of phrases (in line with an endocentric theory of phrasal representation) or whether extra nonterminal information is necessary. If the endocentric hypothesis is true (that is, the representation of a phrase is built from *within* depending on its components but independent of explicit category labels), then the nonterminal types should be easily inferred given the endocentrically-composed representation, and that ablating the nonterminal information would not make much difference in performance. Specifically, we train a GA-RNNG on unlabeled trees (only bracketings without nonterminal types), denoted U-GA-RNNG.

This idea has been explored in research on methods for learning syntax with less complete annotation (Pereira and Schabes, 1992). A key finding from Klein and Manning (2002) was that,

	Noun phrases	Verb phrases	Prepositional phrases
1	Canadian (0.09) Auto (0.31) Workers (0.2) union (0.22) president (0.18)	buying (0.31) and (0.25) selling (0.21) NP (0.23)	ADVP (0.14) on (0.72) NP (0.14)
2	no (0.29) major (0.05) Eurobond (0.32) or (0.01) foreign (0.01) bond (0.1) offerings (0.22)	ADVP (0.27) show (0.29) PRT (0.23) PP (0.21)	ADVP (0.05) for (0.54) NP (0.40)
3	Saatchi (0.12) client (0.14) Philips (0.21) Lighting (0.24) Co. (0.29)	pleaded (0.48) ADJP (0.23) PP (0.15) PP (0.08) PP (0.06)	ADVP (0.02) because (0.73) of (0.18) NP (0.07)
4	nonperforming (0.18) commercial (0.23) real (0.25) estate (0.1) assets (0.25)	received (0.33) PP (0.18) NP (0.32) PP (0.17)	such (0.31) as (0.65) NP (0.04)
5	the (0.1) Jamaica (0.1) Tourist (0.03) Board (0.17) ad (0.20) account (0.40)	cut (0.27) NP (0.37) PP (0.22) PP (0.14)	from (0.39) NP (0.49) PP (0.12)
6	the (0.0) final (0.18) hour (0.81)	to (0.99) VP (0.01)	of (0.97) NP (0.03)
7	their (0.0) first (0.25) test (0.77)	were (0.77) n't (0.22) VP (0.01)	in (0.93) NP (0.07)
8	Apple (0.62) , (0.02) Compaq (0.1) and (0.01) IBM (0.25)	did (0.39) n't (0.60) VP (0.01)	by (0.96) S (0.04)
9	both (0.02) stocks (0.03) and (0.06) futures (0.88)	handle (0.09) NP (0.91)	at (0.99) NP (0.01)
10	NP (0.01) , (0.0) and (0.98) NP (0.01)	VP (0.15) and (0.83) VP (0.02)	NP (0.1) after (0.83) NP (0.06)

Table 5: Attention weight vectors for some representative samples for NPs, VPs, and PPs.

given bracketing structure, simple dimensionality reduction techniques could reveal conventional nonterminal categories with high accuracy; Petrov et al. (2006) also showed that latent variables can be used to recover fine-grained nonterminal categories. We therefore expect that the vector embeddings of the constituents that the U-GA-RNNG correctly recovers (on test data) will capture categories similar to those in the Penn Treebank.

Experiments. Using the same hyperparameters and the PTB dataset, we first consider *unlabeled* F_1 parsing accuracy. On test data (with the usual split), the GA-RNNG achieves 94.2%, while the U-GA-RNNG achieves 93.5%. This result suggests that nonterminal category labels add a relatively small amount of information compared to purely endocentric representations.

Visualization. If endocentricity is largely sufficient to account for the behavior of phrases, where do our robust intuitions for syntactic category types come from? We use t-SNE (van der Maaten and Hinton, 2008) to visualize composed phrase vectors from the U-GA-RNNG model applied to the unseen test data. Fig. 4 shows that the U-GA-RNNG tends to recover nonterminal categories as encoded in the PTB, even when trained without them.⁸ These results suggest nonterminal types can be inferred from the purely endocentric compositional process to a certain extent, and that the phrase clusters found by the U-GA-RNNG largely overlap with nonterminal categories.

Analysis of PP and SBAR. Figure 4 indicates a certain degree of overlap between SBAR (red) and PP (yellow). As both categories are interesting from the linguistic perspective and quite similar, we visualize the learned phrase vectors of 40 randomly selected SBARs and PPs from the test set (using U-GA-RNNG), illustrated in Figure 5. First, we can see that phrase representations for PPs and SBARs depend less on the nonterminal

⁸We see a similar result for the non-ablated GA-RNNG model, not shown for brevity.



Figure 4: t-SNE on composed vectors when training without nonterminal categories. Vectors in dark blue are VPs, red are SBARs, yellow are PPs, light blue are NPs, and green are Ss.

categories⁹ and more on the connector. For instance, the model learns to cluster phrases that start with words that can be either prepositions or complementizers (e.g., *for*, *at*, *to*, *under*, *by*), regardless of whether the true nonterminal labels are PPs or SBARs. This suggests that SBARs that start with “prepositional” words are similar to PPs from the model’s perspective.

Second, the model learns to disregard the word *that*, as “SBAR \rightarrow *that* S” and “SBAR \rightarrow S” are close together. This finding is intuitive, as complementizer *that* is often optional (Jaeger, 2010), unlike prepositional words that might describe relative time and location. Third, certain categories of PPs and SBARs form their own separate clusters, such as those that involve the words *because* and *of*. We attribute these distinctions to the fact that these words convey different meanings than many prepositional words; the word *of* indicates possession while *because* indicates cause-and-effect relationship. These examples show that, to a certain extent, the GA-RNNG is able to learn non-

⁹Recall that U-GA-RNNG is trained without access to the nonterminal labels; training the model with nonterminal information would likely change the findings.

trivial semantic information, even when trained on a fairly small amount of syntactic data.

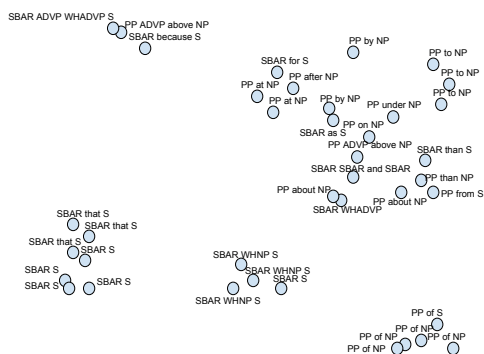


Figure 5: Sample of PP and SBAR phrase representations.

7 Related Work

The problem of understanding neural network models in NLP has been previously studied for sequential RNNs (Karpathy et al., 2015; Li et al., 2016). Shi et al. (2016) showed that sequence-to-sequence neural translation models capture a certain degree of syntactic knowledge of the source language, such as voice (active or passive) and tense information, as a by-product of the translation objective. Our experiment on the importance of composition function was motivated by Vinyals et al. (2015) and Wiseman and Rush (2016), who achieved competitive parsing accuracy without explicit composition. In another work, Li et al. (2015) investigated the importance of recursive tree structures (as opposed to linear recurrent models) in four different tasks, including sentiment and semantic relation classification. Their findings suggest that recursive tree structures are beneficial for tasks that require identifying long-range relations, such as semantic relationship classification, with no conclusive advantage for sentiment classification and discourse parsing. Through the stack-only ablation we demonstrate that the RNNG composition function is crucial to obtaining state-of-the-art parsing performance.

Extensive prior work on phrase-structure parsing typically employs the probabilistic context-free grammar formalism, with lexicalized (Collins, 1997) and nonterminal (Johnson, 1998; Klein and Manning, 2003) augmentations. The conjecture that fine-grained nonterminal rules and labels can be discovered given weaker bracketing structures was based on several studies (Chiang

and Bikel, 2002; Klein and Manning, 2002; Petrov et al., 2006).

In a similar work, Sangati and Zuidema (2009) proposed entropy minimization and greedy familiarity maximization techniques to obtain lexical heads from labeled phrase-structure trees in an unsupervised manner. In contrast, we used neural attention to obtain the “head rules” in the GARNNG; the whole model is trained end-to-end to maximize the log probability of the correct action given the history. Unlike prior work, GARNNG allows the attention weight to be divided among phrase constituents, essentially propagating (weighted) headedness information from multiple components.

8 Conclusion

We probe what recurrent neural network grammars learn about syntax, through ablation scenarios and a novel variant with a gated attention mechanism on the composition function. The composition function, a key differentiator between the RNNG and other neural models of syntax, is crucial for good performance. Using the attention vectors we discover that the model is learning something similar to heads, although the attention vectors are not completely peaked around a single component. We show some cases where the attention vector is divided and measure the relationship with existing head rules. RNNGs without access to nonterminal information during training are used to support the hypothesis that phrasal representations are largely endocentric, and a visualization of representations shows that traditional nonterminal categories fall out naturally from the composed phrase vectors. This confirms previous conjectures that bracketing annotation does most of the work of syntax, with nonterminal categories easily discoverable given bracketing.

Acknowledgments

This work was sponsored in part by the Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O) under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114; it was also supported in part by Contract No. W911NF-15-1-0543 with DARPA and the Army Research Office (ARO). Approved for public release, distribution unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.
- David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proc. of COLING*.
- Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proc. of EMNLP*.
- Noam Chomsky. 1970. Remarks on nominalization. In *Readings in English Transformational Grammar*.
- Noam Chomsky. 1993. *A Minimalist Program for Linguistic Theory*.
- Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of EACL*.
- Marie-Catherine De Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual.
- Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*.
- Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. arXiv:1611.01734.
- Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proc. of NAACL*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.
- Ray Jackendoff. 1977. *X' Syntax*.
- T. Florian Jaeger. 2010. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology*, 61(1).
- Janne Bondi Johannessen. 1998. *Coordination*.
- Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of NODALIDA*.
- Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*.
- Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. arXiv:1506.02078.
- Edward L. Keenan. 1987. Multiply-headed noun phrases. *Linguistic Inquiry*, 18(3):481–490.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*.
- Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proc. of ACL*.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.
- Lingpeng Kong and Noah A. Smith. 2014. An empirical comparison of parsing methods for stanford dependencies. arXiv:1404.4314.
- Jiwei Li, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proc. of EMNLP*.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proc. of NAACL*.
- Tom M. Mitchell. 1980. The need for biases in learning generalizations.
- Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*.
- Federico Sangati and Willem Zuidema. 2009. Unsupervised methods for head assignments. In *Proc. of EACL*.
- Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proc. of EMNLP*.
- Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9.
- Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.
- Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of EMNLP*.

Incremental Discontinuous Phrase Structure Parsing with the GAP Transition

Maximin Coavoux^{1,2} and Benoît Crabbé^{1,2,3}

¹Univ. Paris Diderot, Sorbonne Paris Cité

²Laboratoire de linguistique formelle (LLF, CNRS)

³Institut Universitaire de France

maximin.coavoux@etu.univ-paris-diderot.fr

benoit.crabbe@linguist.univ-paris-diderot.fr

Abstract

This article introduces a novel transition system for discontinuous lexicalized constituent parsing called SR-GAP. It is an extension of the shift-reduce algorithm with an additional gap transition. Evaluation on two German treebanks shows that SR-GAP outperforms the previous best transition-based discontinuous parser (Maier, 2015) by a large margin (it is notably twice as accurate on the prediction of discontinuous constituents), and is competitive with the state of the art (Fernández-González and Martins, 2015). As a side contribution, we adapt span features (Hall et al., 2014) to discontinuous parsing.

1 Introduction

Discontinuous constituent trees can be used to model directly certain specific linguistic phenomena, such as extraposition, or more broadly to describe languages with some degree of word-order freedom. Although these phenomena are sometimes annotated with indexed traces in CFG treebanks, other constituent treebanks are natively annotated with discontinuous constituents, e.g. the Tiger corpus (Brants, 1998).

From a parsing point of view, discontinuities pose a challenge. Mildly context-sensitive formalisms, that are expressive enough to model discontinuities have high parsing complexity. For example, the CKY algorithm for a binary probabilistic LCFRS is in $\mathcal{O}(n^{3k})$, where k is the fan-out of the grammar (Kallmeyer, 2010).

Recently, there have been several proposals for direct discontinuous parsing. They correspond roughly to three different parsing paradigms. (i) Chart parsers are based on probabilistic LCFRS (Kallmeyer and Maier, 2013; Maier, 2010; Evang

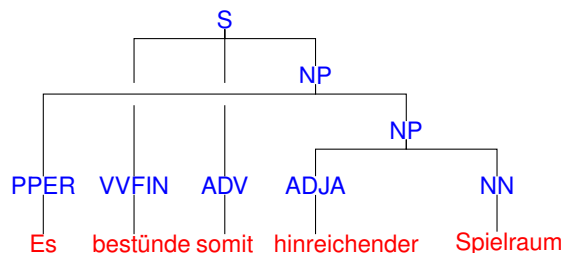


Figure 1: Discontinuous tree extracted from the Tiger corpus (punctuation removed).

and Kallmeyer, 2011), or on the Data-Oriented Parsing (DOP) framework (van Cranenburgh, 2012; van Cranenburgh and Bod, 2013; van Cranenburgh et al., 2016). However, the complexity of inference in this paradigm requires to design elaborate search strategies and heuristics to make parsing run-times reasonable. (ii) Several approaches are based on modified non-projective dependency parsers, for example Hall and Nivre (2008), or more recently Fernández-González and Martins (2015) who provided a surprisingly accurate parsing method that can profit from efficient dependency parsers with rich features. (iii) Transition-based discontinuous parsers are based on the easy-first framework (Versley, 2014a) or the shift-reduce algorithm augmented with a swap action (Maier, 2015). In the latter system, which we will refer to as SR-SWAP, a SWAP action pushes the second element of the stack back onto the buffer.

Although SR-SWAP is fast and obtained good results, it underperforms Fernández-González and Martins (2015)'s parser by a large margin. We believe this result does not indicate a fatal problem for the transition-based framework for discontinuous parsing, but emphasizes several limitations inherent to SR-SWAP, in particular the length of derivations (Section 3.5).

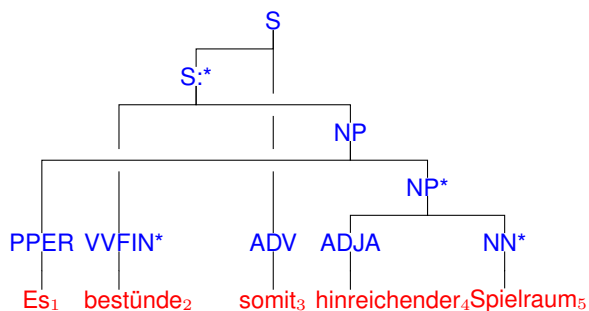


Figure 2: Lexicalized binarized tree. The symbol ‘*’ encodes head information. Symbols suffixed by ‘:’ are temporary symbols introduced by the binarization.

Contributions We introduce a novel transition system for discontinuous parsing we call SR-GAP. We evaluate this algorithm on two German treebanks annotated with discontinuous constituents, and show that, in the same experimental settings, it outperforms the previous best transition-based parser of Maier (2015), and matches the best published results on these datasets (Fernández-González and Martins, 2015). We provide a theoretical and empirical comparison between SR-GAP and SR-SWAP. Finally we adapt span-based features to discontinuous parsing.

The code and preprocessing scripts to reproduce experiments are available for download at <https://github.com/mcoavoux/mtg>.

2 Discontinuous Shift-Reduce Parsing

In this section we present SR-GAP, a transition system for discontinuous constituent parsing. SR-GAP is an extension of the shift-reduce system. In what follows, we assume that part-of-speech tags for the words of a sentence are given. A *terminal* refers to a tag-word couple.

2.1 Standard Shift-Reduce Constituent Parsing

The shift-reduce system is based on two data structures. The stack (S) contains tree fragments representing partial hypotheses and the buffer (B) contains the remaining terminals. A parsing configuration is a couple $\langle S, B \rangle$. Initially, B contains the sentence as a list of terminals and S is empty.

The three types of actions are defined as follows.

- $\text{SHIFT}(\langle S, w|B \rangle) = \langle S|w, B \rangle$
- $\text{REDUCE-X}(\langle S|A_1|A_2, B \rangle) = \langle S|X, B \rangle$

- $\text{REDUCEUNARY-X}(\langle S|A, B \rangle) = \langle S|X, B \rangle$

where X is any non-terminal in the grammar. The analysis terminates when the buffer is empty and the only symbol in the stack is an axiom of the grammar. This transition system can predict any labelled constituent tree over a set of non-terminal symbols N .

These three action types can only produce binary trees. In practice, shift-reduce parsers often assume that their data are binary. In this article, we assume that trees are binary, that each node X is annotated with a head h (notation: $X[h]$), and that the only unary nodes are parents to a terminal. Therefore, we only need unary reductions immediately after a SHIFT. We refer the reader to Section 3.1 for the description of the preprocessing operations.

2.2 SR-GAP Transition System

In order to handle discontinuous constituents, we need an algorithm expressive enough to predict non-projective trees.

Compared to the standard shift-reduce algorithm, the main intuition behind SR-GAP is that reductions do not always apply to the two top-most elements in the stack. Instead, the left element of a reduction can be any element in the stack and must be chosen dynamically.

To control the choice of the symbols to which a reduction applies, the usual stack is split into two data structures. A deque D represents its top and a stack S represents its bottom. Alternatively, we could see these two data structures as a single stack with two pointers indicating its top and a split point. The respective top-most element of D and S are those available for a reduction.

The transition system is given as a deductive system in Figure 3. A REDUCE-X action pops the top element of S and the top element of D , flushes the content of D to S and finally pushes a new non-terminal X on D . As feature extraction (Section 3.1) relies on the lexical elements, we use two types of binary reductions, left and right, to assign heads to new constituents. Unary reductions replace the top of D by a new non-terminal.

The SHIFT action flushes D to S , pops the next token from B and pushes it onto D .

Finally, the GAP action pops the first element of S and appends it at the bottom of D . This action enables elements below the top of S to be also available for a reduction with the top of D .

Input	$t_1[w_1]t_2[w_2] \dots t_n[w_n]$
Axiom	$\langle \epsilon, \epsilon, t_1[w_1]t_2[w_2] \dots t_n[w_n] \rangle$
Goal	$\langle \epsilon, S[w], \epsilon \rangle$
SH	$\frac{\langle S, D, t[w] B \rangle}{\langle S D, t[w], B \rangle}$
RU(X)	$\frac{\langle S, d_0[h], B \rangle}{\langle S, X[h], B \rangle}$
RR(X)	$\frac{\langle S s_0[h], D d_0[h'], B \rangle}{\langle S D, X[h'], B \rangle}$
RL(X)	$\frac{\langle S s_0[h], D d_0[h'], B \rangle}{\langle S D, X[h], B \rangle}$
GAP	$\frac{\langle S s_0[h], D, B \rangle}{\langle S, s_0[h] D, B \rangle}$

Figure 3: SR-GAP transition system for discontinuous phrase structure parsing. $X[h]$ denotes a non-terminal X and its head h . s_0 and d_0 denote the top-most elements of respectively S and D .

Constraints In principle, a tree can be derived by several distinct sequences of actions. If a SHIFT follows a sequence of GAPS, the GAPS will have no effect, because SHIFT flushes D to S before pushing a new terminal to D . In order to avoid useless GAPS, we do not allow a SHIFT to follow a GAP. A GAP must be followed by either another GAP or a binary reduction.

Moreover, as we assume that preprocessed trees do not contain unary nodes, except possibly above the terminal level, unary reductions are only allowed immediately after a SHIFT. Other constraints on the transition system are straightforward, we refer the reader to Table 7 of Appendix A for the complete list.

2.3 Oracle and Properties

Preliminary Definitions Following Maier and Lichte (2016), we define a discontinuous tree as a rooted connected directed acyclic graph $T = (V, E, r)$ where

- V is a set of nodes;
- $r \in V$ is the root node;
- $E : V \times V$ is a set of (directed) edges and E^* is the reflexive transitive closure of E .

If $(u, v) \in E$, then u is the parent of v . Each node has a unique parent (except the root that has none). Nodes without children are *terminals*.

The *right index* (resp. *left index*) of a node is the index of the rightmost (resp. leftmost) terminal dominated by this node. For example, the left index of the node labelled S : in Figure 2 is 1 and its right index is 5.

Oracle We extract derivations from trees by following a simple tree traversal. We start with an initial configuration. While the configuration is not final, we derive a new configuration by performing the gold action, which is chosen as follows:

- if the nodes at the top of S and at the top of D have the same parent node in the gold tree, perform a reduction with the parent node label;
- if the node at the top of D and the i^{th} node in S have the same parent node, perform $i - 1$ GAP;
- otherwise, perform a SHIFT, optionally followed by a unary reduction in the case where the parent node of the top of D has only one child.

For instance, the gold sequence of actions for the tree in Figure 2 is the sequence SH, SH, SH, SH, SH, RR(NP), GAP, GAP, RR(NP), GAP, RL(S:), RR(S). Table 1 details the sequence of configurations obtained when deriving this tree.

Given the constraints defined in Section 2.2, and if we ignore lexicalisation, there is a bijection between binary trees and derivations.¹ To see why, we define a total order $<$ on the nodes of a tree. Let n and n' be two nodes and let $n < n'$ iff either $rindex(n) < rindex(n')$ or $(n', n) \in E^*$.

It is immediate that if $(n', n) \in E^*$, then n must be reduced before n' in a derivation. An invariant of the GAP transition system is that the right index of the first element of D is equal to the index of the last shifted element. Therefore, after having shifted the terminal j , it is impossible to create nodes whose right-index is strictly smaller to j . We conclude that during a derivation, the nodes must be created according to the strict total order $<$ defined above. In other words, for a given tree, there is a unique possible derivation which enforces the constraints described above. Recip-

¹But several binarized trees can correspond to the same n -ary tree.

<i>S</i>	<i>D</i>	<i>B</i>	Action
		Es bestünde somit hinreichender Spielraum	
Es	Es	bestünde somit hinreichender Spielraum	SH
Es bestünde	bestünde	somit hinreichender Spielraum	SH
Es bestünde somit	somit	hinreichender Spielraum	SH
Es bestünde somit hinreichender	hinreichender	Spielraum	SH
Es bestünde somit	Spielraum		SH
Es bestünde somit	NP[Spielraum]		RR(NP)
Es bestünde	somit NP[Spielraum]		GAP
Es	bestünde somit NP[Spielraum]		GAP
bestünde somit	NP[Spielraum]		RR(NP)
bestünde	somit NP[Spielraum]		GAP
somit	S:[bestünde]		RL(S:)
	S[bestünde]		RR(S)

Table 1: Example derivation for the sentence in Figure 2, part-of-speech tags are omitted.

roccally, a well-formed derivation corresponds to a unique tree.

Completeness and Soundness The GAP transition system is sound and complete for the set of discontinuous binary trees labelled with a set of non-terminal symbols. When augmented with certain constraints to make sure that predicted trees are unbinarizable (see Table 7 of Appendix A), this result also holds for the set of discontinuous n -ary trees (modulo binarization and unbinarization).

Completeness follows immediately from the correctness of the oracle, which corresponds to a tree traversal in the order specified by $<$.

To prove soundness, we need to show that any valid derivation sequence produces a discontinuous binary tree. It holds from the transition system that no node can have several parents, as parent assignment via REDUCE actions pops the children nodes and makes them unavailable to other reductions. This implies that at any moment, the content of the stack is a forest of discontinuous trees. Moreover, at each step, at least one action is possible (thanks to the constraints on actions). As there can be no cycles, the number of actions in a derivation is upper-bounded by $\frac{1}{2}(n^2 + n)$ for a sentence of length n (see Appendix A.1). Therefore, the algorithm can always reach a final configuration, where the forest only contains one discontinuous tree.

The correctness of SR-GAP system holds only for the robust case: that is the full set of labeled discontinuous trees, and not, say, for the set of trees derived by a true LCFRS grammar also able to reject agrammatical sentences. From an empirical point of view, a transition system that over-

generates is necessary for robustness, and is desirable for fast approximate linear-time inference. However, from a formal point of view, the relationship of the SR-GAP transition system with automata explicitly designed for LCFRS parsing (Villemonde de La Clergerie, 2002; Kallmeyer and Maier, 2015) requires further investigations.

2.4 Length of Derivations

Any derivation produced by SR-GAP for a sentence of length n will contain exactly n SHIFTS and $n - 1$ binary reductions. In contrast, the number of unary reductions and GAP actions can vary. Therefore several possible derivations for the same sentence may have different lengths.

This is a recurring problem for transition-based parsing because it undermines the comparability of derivation scores. In particular, Crabbé (2014) observed that the score of a parse item is approximately linear in the number of previous transitions, which creates a bias towards long derivations.

Different strategies have been proposed to ensure that all derivations have the same length (Zhu et al., 2013; Crabbé, 2014; Mi and Huang, 2015). Following Zhu et al. (2013), we use an additional IDLE action that can only be performed when a parsing item is final. Thus, short derivations are padded until the last parse item in the beam is final. IDLE actions are scored exactly like any other action.

SR-CGAP As an alternative strategy to the problem of comparability of hypotheses, we also present a variant of SR-GAP, called SR-CGAP, in which the length of any derivation only depends on the length of the sentence. In SR-CGAP, each

SHIFT action must be followed by either a unary reduction or a ghost reduction (Crabbé, 2015), and each binary reduction must be preceded by exactly one compound GAP_i action ($i \in \{0, \dots, m\}$) specifying the number i of consecutive standard GAPS. For example, GAP_0 will have no effect, and GAP_2 counts as a single action equivalent to two consecutive GAPS. We call these actions COMPOUNDGAP, following the COMPOUNDSWAP actions of Maier (2015).

With this set of actions, any derivation will have exactly $4n - 2$ actions, consisting of n shifts, n unary reductions or ghost reductions, $n - 1$ compound gaps, and $n - 1$ reductions.

The parameter m (maximum index of a compound gap) is determined by the maximum number of consecutive gaps observed in the training set. Contrary to SR-GAP, SR-CGAP is not complete, as some discontinuous trees whose derivation should contain more than m consecutive GAPS cannot be predicted.

2.5 Beam Search with a Tree-structured Stack

A naive beam implementation of SR-GAP will copy the whole parsing configuration at each step and for each item in the beam. This causes the parser algorithm to have a practical $\mathcal{O}(k \cdot n^2)$ complexity, where k is the size of the beam and n the length of a derivation. To overcome this, one can use a tree-structured stack (TSS) to factorize the representations of common prefixes in the stack as described by (Goldberg et al., 2013) for projective dependency parsing. However the discontinuities entail that a limited amount of copying cannot be entirely avoided. When a reduction follows n GAP actions, we need to grow a new branch of size $n + 1$ in the tree-structured stack to account for reordering. The complexity of the inference becomes $\mathcal{O}(k \cdot (n + g))$ where g is the number of gaps in the derivation. As there are very few gap actions (in proportion) in the dataset, the practical runtime is linear in the length of the derivation.

2.6 Relationship to Dependency Parsing Algorithms

The transition system presented in this article uses two distinct data structures to represent the stack. In this respect, it belongs to the family of algorithms presented by Covington (2001) for dependency parsing. Covington’s algorithm iterates over every possible pair of words in a sentence

and decides for each pair whether to attach them – with a left or right arc – or not. This algorithm can be formulated as a transition system with a split stack (Gómez-Rodríguez and Fernández-González, 2015).

3 Experiments

3.1 Datasets

We evaluated our model on two corpora, namely the Negra corpus (Skut et al., 1997) and the Tiger corpus (Brants, 1998). To ensure comparability with previous work, we carried out experiments on several instantiations of these corpora.

We present results on two instantiations of Negra. NEGRA-30 consists of sentences whose length is smaller than, or equal to, 30 words. We used the same split as Maier (2015). A second instantiation, NEGRA-ALL, contains all the sentences of the corpus, and uses the standard split (Dubey and Keller, 2003).

For the Tiger corpus, we also use two instantiations. TIGERHN08 is the split used by Hall and Nivre (2008). TIGERM15 is the split of Maier (2015), which corresponds to the SPMRL split (Seddah et al., 2013).² We refer the reader to Table 8 in Appendix A for further details on the splits used.

For both corpora, the first step of preprocessing consists in removing function labels and reattaching the nodes attached to the ROOT and causing artificial discontinuities (these are mainly punctuation terminals).³

Then, corpora are head-annotated using the headrules included in the DISCO-DOP package, and binarized by an order-0 head-Markovization (Klein and Manning, 2003). There is a rich literature on binarizing LCFRS (Gómez-Rodríguez et al., 2009; Gildea, 2010), because both the gap-degree and the rank of the resulting trees need to be minimized in order to achieve a reasonable complexity when using chart-based parsers (Kallmeyer and Maier, 2013). However, this seems not to be a problem for transition-based parsing, and the gains of using optimized binarization algorithms do not seem to be worth the

²As in previous work (Maier, 2015), two sentences (number 46234 and 50224) are excluded from the test set because they contain annotation errors.

³We used extensively the publicly available software TREETOOLS and DISCO-DOP for these preprocessing steps. The preprocessing scripts will be released with the parser source code for full replicability.

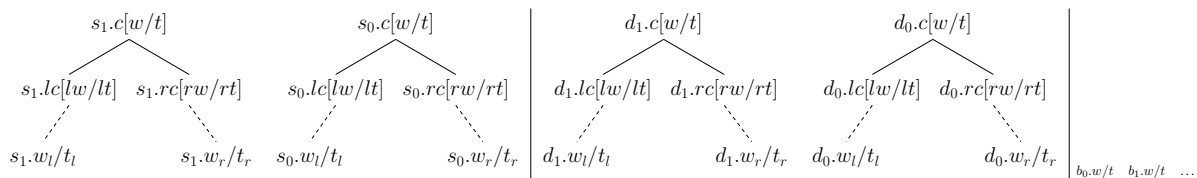


Figure 4: Schematic representation of the top-most elements of S , D and B , using the notations introduced in Table 2. Due to discontinuities, it is possible that both the left- and right- index of s_i are generated by the same child of s_i .

BASELINE				
b_0tw	b_1tw	b_2tw	b_3tw	d_0tc
d_0wc	s_0tc	s_0wc	s_1tc	s_1wc
s_2tc	s_2wc	s_0lwc	s_0rwc	d_0lwc
d_0rwc	s_0wd_0w	s_0wd_0c	s_0cd_0w	s_0cd_0c
b_0wd_0w	b_0td_0w	b_0wd_0c	b_0td_0c	b_0ws_0w
b_0ts_0w	b_0ws_0c	b_0ts_0c	b_0wb_1w	b_0wb_1t
b_0tb_1w	b_0tb_1t	$s_0cs_1wd_0c$	$s_0cs_1cd_0c$	$b_0ws_0cd_0c$
$b_0ts_0cd_0c$	$b_0ws_0wd_0c$	$b_0ts_0wd_0c$	$s_0cs_1cd_0w$	$b_0ts_0cd_0w$
+ EXTENDED				
s_3tc	s_3wc	s_1lwc	s_1rwc	d_1tc
d_1wc	d_2tc	d_2wc	$s_0cs_1cd_0c$	$s_2cs_0cs_1cd_0c$
$s_0cd_1cd_0c$	$s_0cd_1cs_1cd_0c$			
+ SPANS				
$d_0cw_1w_r$	$s_0cw_1w_r$	$d_0cw_1s_0w_r$	$d_0cw_r s_0w_l$	$d_0w_1w_r b_0w$
$d_0w_1w_r b_1w$	$d_0cw_r s_0w_{lo}$	$d_0ct_1w_r$	$d_0cw_{lt_r}$	$d_0ct_{lt_r}$
$s_0ct_1w_r$	$s_0cw_{lt_r}$	$s_0ct_{lt_r}$	$d_0ct_1s_0w_r$	$d_0cw_1s_0t_r$
$d_0ct_1s_0t_r$	$d_0ct_r s_0w_l$	$d_0cw_r s_0t_l$	$d_0ct_r s_0t_l$	$d_0w_1w_r b_0t$
$d_0w_1w_r b_1t$	d_0cw_{lo}	d_0ct_{lo}	s_0cw_{ro}	s_0ct_{ro}

Table 2: Feature templates. s , d and b refer respectively to the data structures (S , D , B) presented in Section 2.2. The integers are indices on these data structures. *left* and *right* refer to the children of nodes. We use c , w and t to denote a node’s label, its head and the part-of-speech tag of its head. When used as a subscript, l (r) refers to the left (right) index of a node. Finally lo (ro) denotes the token immediately left to the left index (right to the right index). See Figure 4 for a representation of a configuration with these notations.

complexity of these algorithms (van Cranenburgh et al., 2016).

Unless otherwise indicated, we did experiments with gold part-of-speech tags, following a common practice for discontinuous parsing.

3.2 Classifier

We used an averaged structured perceptron (Collins, 2002) with early-update training (Collins and Roark, 2004). We use the hash trick (Shi et al., 2009) to speed up feature hashing. This has no noticeable effect on accuracy and this improves training and parsing speed. The only hyperparameter of the perceptron is the number of training epochs.

We fixed it at 30 for every experiment, and shuffled the training set before each epoch.

Features We tested three feature sets described in Table 2 and Figure 4. The BASELINE feature set is the transposition of Maier (2015)’s baseline features to the GAP transition system. It is based on B , on S , and on the top element of D , but does not use information from the rest of D (i.e. the gapped elements). This feature set was designed in order to obtain an experimental setting as close as possible to that of Maier (2015).

In contrast, the EXTENDED feature set includes information from further in D , as well as additional context in S and n -grams of categories from both S and D .

The third feature set SPANS is based on the idea that constituent boundaries contain critical information (Hall et al., 2014) for phrase structure parsing. This intuition is also confirmed in the context of lexicalized transition-based constituent parsing (Crabbé, 2015). To adapt this type of features to discontinuous parsing, we only rely on the right and left index of nodes, and on the tokens preceding the left index or following the right index.⁴

Unknown Words In order to learn parameters for unknown words and limit feature sparsity, we replace hapaxes in the training set by an UNKNOWN pseudo-word. This accounts for an improvement of around 0.5 F1.

3.3 Results

We report results on test sets in Table 3. All the metrics were computed by DISCO-DOP with the parameters included in this package (`proper.prm`). The metrics are labelled F1, ignoring roots and punctuation. We also present metrics which consider only the discontinuous constituents (Disc. F1 in Tables 3 and 4), as these

⁴For discontinuous constituents, internal boundaries (for gaps) might prove useful too.

Method	NEGRA-30	NEGRA-ALL		TIGERHN08		TIGERM15	
	All	$L \leq 40$	All	$L \leq 40$	All	SPMRL / standard	
Fernández-González and Martins (2015)	dep2const	82.56†	81.08	80.52	85.53	84.22	80.62 / -
Hall and Nivre (2008)	dep2const	-	-	-	79.93	-	-/-
van Cranenburgh (2012)	DOP	-	72.33	71.08	-	-	-/-
van Cranenburgh and Bod (2013)	DOP	-	76.8	-	-	-	-/-
Kallmeyer and Maier (2013)	LCFRS	75.75	-	-	-	-	-/-
Versley (2014a)	EAFI	-	-	-	74.23	-	-/-
Maier (2015) (baseline, b=(Ne=8/Ti=4))	SR-SWAP	75.17 (15.76)	-	-	-	-	-/-
Maier (2015) (best, b=(Ne=8/Ti=4))	SR-SWAP	76.95 (19.82)	-	-	79.52	-	- / 74.71 (18.77)
Maier and Lichte (2016) (best, b=4)	SR-SWAP	-	-	-	80.02	-	- / 76.46 (16.31)
This work, beam=4		F1 (Disc. F1)					
GAP, BASELINE	SR-GAP	79.31 (38.66)	79.29 (39.78)	78.53 (38.64)	82.84 (47.13)	81.67 (44.83)	78.77 / 78.86 (41.36)
GAP, +EXTENDED	SR-GAP	80.44 (41.13)	80.34 (43.42)	79.79 (43.56)	83.57 (50.91)	82.43 (48.81)	79.42 / 79.51 (43.76)
GAP, +SPANS	SR-GAP	81.64 (42.94)	81.70 (47.17)	81.28 (46.85)	84.40 (51.98)	83.16 (49.76)	80.30 / 80.40 (46.50)
CGAP, BASELINE	SR-CGAP	79.61 (41.06)	79.32 (43.49)	78.64 (42.13)	82.90 (47.86)	81.68 (45.55)	78.32 / 78.41 (39.99)
CGAP, +EXTENDED	SR-CGAP	80.26 (40.52)	80.48 (43.42)	79.98 (42.60)	83.23 (50.57)	82.00 (48.28)	79.32 / 79.42 (44.66)
CGAP, +SPANS	SR-CGAP	81.16 (42.39)	81.41 (44.73)	80.89 (44.13)	83.92 (50.83)	82.79 (48.84)	80.38 / 80.48 (46.17)
This work, beam=32		F1 (Disc. F1)					
GAP, BASELINE	SR-GAP	80.57 (42.16)	80.20 (43.87)	79.75 (42.80)	83.53 (51.91)	82.41 (49.63)	79.60 / 79.69 (44.77)
GAP, +EXTENDED	SR-GAP	81.61 (45.75)	81.13 (47.52)	80.54 (46.89)	84.33 (53.84)	83.17 (51.88)	80.50 / 80.59 (46.45)
GAP, +SPANS	SR-GAP	82.46 (47.35)	82.76 (51.82)	82.16 (50.00)	85.11 (55.99)	84.01 (54.26)	81.50 / 81.60 (49.17)

Table 3: Final test results. For TIGERM15, we report metrics computed with the SPMRL shared task parameters (see Section 3.3), as well as the standard parameters. †Trained on NEGRA-ALL.

can give some qualitative insight into the strengths and weaknesses of our model.

For experiments on TIGERM15, we additionally report evaluation scores computed with the SPMRL shared task parameters⁵ for comparability with previous work.

SR-GAP vs SR-CGAP In most experimental settings, SR-CGAP slightly underperformed SR-GAP. This result came as a surprise, as both compound actions for discontinuities (Maier, 2015) and ghost reductions (Crabbé, 2014) were reported to improve parsing.

We hypothesize that this result is due to the rarity of unary constituents in the datasets and to the difficulty to predict COMPOUNDGAPS with a bounded look at D and S caused by our practical definition of feature templates (Table 2). In contrast, predicting gaps separately involves feature extraction at each step, which crucially helps.

Feature Sets The EXTENDED feature set outperforms the baseline by up to one point of F1. This emphasizes that information about gapped non-terminal is important. The SPANS feature set gives another 1 point improvement. This demonstrates clearly the usefulness of span features for discontinuous parsing. A direct extension of this feature set would include information about the

⁵These are included in http://pauillac.inria.fr/~seddah/evalb_spmrl2013.tar.gz. In this setting, we reattach punctuation to the root node before evaluation.

Beam size	TIGERHN8		TIGERM15	
	F1	Disc. F1	F1	Disc. F1
2	81.86	48.49	84.28	49.04
4	83.27	53.00	85.43	53.14
8	83.61	54.42	85.93	55.00
16	83.84	54.81	86.13	56.17
32	84.32	56.22	86.10	55.50
64	84.14	56.01	86.30	56.90
128	84.05	55.76	86.13	57.04

Table 4: Results on development sets for different beam sizes.

boundaries of gaps in a discontinuous constituent. A difficulty of this extension is that the number of gaps in a constituent can vary.

3.4 Comparisons with Previous Works

There are three main approaches to direct discontinuous parsing.⁶ One such approach is based on unprojective or pseudo-projective dependency parsing (Hall and Nivre, 2008; Fernández-González and Martins, 2015), and aims at enriching dependency labels in such a way that constituents can be retrieved from the dependency tree. The advantage of such systems is that they can use off-the-shelf dependency parsers with rich features and efficient inference.

⁶As opposed to non-direct strategies, based for example on PCFG parsing and post-processing.

The second approach is chart-based parsing, as exemplified by the DOP (Data Oriented Parsing) models of van Cranenburgh (2012) and van Cranenburgh et al. (2016) and Probabilistic LCFRS (Kallmeyer and Maier, 2013; Evang and Kallmeyer, 2011).

The last paradigm is transition-based parsing. Versley (2014a) and Versley (2014b) use an easy-first strategy with a swap transition. Maier (2015) and Maier and Lichte (2016) use a shift-reduce algorithm augmented with a swap transition.

Table 3 includes recent results from these various parsers. The most successful approach so far is that of Fernández-González and Martins (2015), which outperforms by a large margin transition-based parsers (Maier, 2015; Maier and Lichte, 2016).

SR-GAP vs SR-SWAP In the same settings (baseline features and beam size of 4), SR-GAP outperforms SR-SWAP by a large margin on all datasets. It is also twice as accurate when we only consider discontinuous constituents.

In Section 3.5, we analyse the properties of both transition systems and give hypotheses for the performance difference.

Absolute Scores On all datasets, our model reaches or outperforms the state of the art (Fernández-González and Martins, 2015). This is still the case in a more realistic experimental setup with predicted tags, as reported in Table 5.⁷

As pointed out by Maier and Lichte (2016), a limitation of shift-reduce based parsing is the locality of the feature scope when performing the search. The parser could be in states where the necessary information to take the right parsing decision is not accessible with the current scoring model.

To get more insight into this hypothesis, we tested large beam sizes. If the parser maintains a much larger number of hypotheses, we hope that it could compensate for the lack of information by delaying certain decisions. In Table 4, we present additional results on development sets of both instantiations of the TIGER corpus, with different beam sizes. As was expected, a larger beam size

⁷Like Fernández-González and Martins (2015), we used the predicted tags provided by the SPMRL shared task organizers.

TIGERM15	F1 (spmrl.prm)	
	≤ 70	All
Versley (2014b)	73.90	-
Fernández-González and Martins (2015)	77.72	77.32
SR-GAP, beam=32, +SPANS	79.44	79.26

Table 5: Results on the Tiger corpus in the SPMRL predicted tag scenario.

gives better results. The beam size controls the tradeoff between speed and accuracy.⁸

Interestingly, the improvement from a larger beam size is greater on discontinuous constituents than overall. For example, from 16 to 32, F1 improves by 0.5 on TIGERHN8 and F1 on discontinuous constituents improves by 1.4.

This suggests that further improvements could be obtained by augmenting the lookahead on the buffer and using features further on S and D . We plan in the future to switch to a neural model such as a bi-LSTM in order to obtain more global representations of the whole data structures (S , D , B).

3.5 Discussion: Comparing SR-SWAP and SR-GAP

This section investigates some differences between SR-SWAP and SR-GAP. We think that characterizing properties of transition systems helps to gain better intuitions into the problems inherent to discontinuous parsing.

Derivation Length Assuming that GAP or SWAP are the hardest actions to predict and are responsible for the variability of the lengths of derivation, we hypothesize that the number of these actions, hence the length of a derivation, is an important factor. Shorter derivations are less prone to error propagation.

In both cases, the shortest possible derivation for a sentence of length n corresponds to a projective tree, as the derivation will not contain any SWAP or GAP.

In the worst case, i.e. the tree that requires the longest derivation to be produced by a transition system, SR-GAP is asymptotically twice as more economical than SR-SWAP (Table 6). In Figure 5 of Appendix A, we present the trees corresponding to the longest possible derivations in both cases.

⁸Training with the full-feature model took approximately 1h30 on TIGERM15 with a beam size of 4. Parsing speed, in the same setting, is approximately 4,700 tokens per second (corresponding to 260 sentences per second) on a single Xeon 2.30 GHz CPU.

	SR-GAP	SR-SWAP	SR-CSWAP
Theoretical longest derivation	$\frac{n^2+n}{2}$	$n^2 - n + 1$	
Longest derivation	276	2187	1276
Total number of gaps/swaps	64096	411970	
Max consecutive gaps/swaps	10	69	
Avg. deriv. length wrt n	$2.03n$	$3.09n$	$2.66n$

Table 6: Statistics on Tiger train corpus. n is the length of a sentence. SR-CSWAP is a variant of SR-SWAP proposed by Maier (2015).

These trees maximise the number of GAP and SWAP actions.

The fact that derivations are shorter with SR-GAP is confirmed empirically. In Table 6, we present several metrics computed on the train section of TIGERM15. In average, SR-SWAP derivations are empirically 50% longer than SR-GAP derivations. Despite handling discontinuities, SR-GAP derivations are not noticeably longer than those we would get with a standard shift-reduce transition system (n shifts and $n - 1$ binary reductions).

Intuitively, the difference in length of derivations comes from two facts. First, swapped terminals are pushed on the buffer and must be shifted once more, whereas with SR-GAP, each token is shifted exactly once. Second, transition systems for discontinuous parsing implicitly predict an order on terminals (discontinuous trees can be transformed to continuous trees by changing the precedence order on terminals). With SR-SWAP, reordering is done by swapping two terminals. In contrast, SR-GAP can swap complex non-terminals (already ordered chunks of terminals), making the reordering more efficient in terms of number of operations.

It would be interesting to see if SR-SWAP is improved when swapping non-terminals is allowed. However, it would make feature extraction more complex, because it would no longer be assumed that the buffer contains only tokens.

The effect of the derivation length is confirmed by Maier and Lichte (2016) who explored different oracles for SR-SWAP and found that oracles producing shorter derivations gave better results.

Feature Locality A second property which explains the performance of SR-GAP is the access to three data structures (vs two for SR-SWAP) for extracting features; SR-GAP has access to an extended domain of locality. Moreover, with SR-SWAP, the semantics of features from the queue

does not make a distinction between swapped tokens and tokens that have not been shifted yet. When the parser needs to predict a long sequence of consecutive swaps, it is hardly in a position to have access to the relevant information. The use of three data structures, along with shorter sequences of GAP actions, seems to alleviate this problem.

4 Conclusion

We have introduced a novel transition system for lexicalized discontinuous parsing. The SR-GAP transition system produces short derivations, compared to SR-SWAP, while being able to derive any discontinuous tree.

Our experiments show that it outperforms the best previous transition system (Maier, 2015) in similar settings and different datasets. Combined with a span-based feature set, we obtained a very efficient parser with state-of-the-art results.

We also provide an efficient C++ implementation of our parser, based on a tree-structured stack.

Direct follow-ups to this work consist in switching to a neural scoring model to improve the representations of D and S and alleviate the locality issues in feature extraction (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016).

Acknowledgments

We thank three anonymous reviewers, as well as Héctor Martínez Alonso, Olga Seminck and Chloé Braud for comments and suggestions to improve prior versions of this article. We also thank Djamel Seddah for help with the SPMRL dataset.

References

- Thorsten Brants. 1998. The NeGra export format for annotated corpora. Technical Report 98, Universität des Saarlandes, Saarbrücken, April.
- Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

- Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.
- Benoit Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *Proceedings of the twenty-fifth International Conference on Computational Linguistics*, Dublin, Ireland, August.
- Benoit Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1847–1856, Lisbon, Portugal, September. Association for Computational Linguistics.
- James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany, August. Association for Computational Linguistics.
- Amit Dubey and Frank Keller. 2003. Probabilistic parsing for german using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan, July. Association for Computational Linguistics.
- Kilian Evang and Laura Kallmeyer. 2011. Plcfrs parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies, IWPT '11*, pages 104–116, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China, July. Association for Computational Linguistics.
- Daniel Gildea. 2010. Optimal parsing strategies for linear context-free rewriting systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 769–776, Los Angeles, California, June. Association for Computational Linguistics.
- Yoav Goldberg, Kai Zhao, and Liang Huang. 2013. Efficient implementation of beam-search incremental parsers. In *ACL (2)*, pages 628–633. The Association for Computer Linguistics.
- Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 256–261, Beijing, China, July. Association for Computational Linguistics.
- Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 539–547, Boulder, Colorado, June. Association for Computational Linguistics.
- Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In Bengt Nordström and Aarne Ranta, editors, *Advances in Natural Language Processing, 6th International Conference, GoTAL 2008, Gothenburg, Sweden, August 25-27, 2008, Proceedings*, volume 5221 of *Lecture Notes in Computer Science*, pages 169–180. Springer.
- David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland, June. Association for Computational Linguistics.
- Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.
- Laura Kallmeyer and Wolfgang Maier. 2015. Lr parsing for lcfrs. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1250–1255, Denver, Colorado, May–June. Association for Computational Linguistics.
- Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer Publishing Company, Incorporated, 1st edition.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association of Computational Linguistics – Volume 4, Issue 1*, pages 313–327.
- Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1, ACL '03*, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 47–57, San

- Diego, California, June. Association for Computational Linguistics.
- Wolfgang Maier. 2010. Direct parsing of discontinuous constituents in german. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66, Los Angeles, CA, USA, June. Association for Computational Linguistics.
- Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China, July. Association for Computational Linguistics.
- Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1030–1035, Denver, Colorado, May–June. Association for Computational Linguistics.
- Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash kernels for structured data. *J. Mach. Learn. Res.*, 10:2615–2637, December.
- Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC.
- Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate dop model. *Proceedings of the International Conference on Parsing Technologies (IWPT 2013)*.
- Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *J. Language Modelling*, 4(1):57–111.
- Andreas van Cranenburgh. 2012. Efficient parsing with linear context-free rewriting systems. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 460–470, Avignon, France, April. Association for Computational Linguistics.
- Yannick Versley. 2014a. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland, August. Dublin City University.
- Yannick Versley. 2014b. Incorporating Semi-supervised Features into Discontinuous Easy-first Constituent Parsing. Technical report, SPMRL Shared Task.
- Éric Villemonte de La Clergerie. 2002. Parsing mildly context-sensitive languages with thread automata. In *Proc. of COLING’02*, August.
- Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *ACL (1)*, pages 434–443. The Association for Computer Linguistics.

A Supplemental Material

Action	Conditions
SHIFT	B is not empty. The last action is not GAP.
GAP	S has at least 2 elements. If d_0 is a temporary symbol, there must be at least one non temporary symbol in S_1 .
RU(X)	The last action is SHIFT. X is an axiom iff this is a one-word sentence.
R(R L)(X)	S is not empty. X is an axiom iff B is empty, and S and D both have exactly one element. If X is a temporary symbol and if B is empty, there must be a non-temporary symbol in either S_1 . or D_1 .
RR(X)	s_0 is not a temporary symbol.
RL(X)	d_0 is not a temporary symbol.
IDLE	The configuration must be final, i.e. S and B are empty and the only element of S is the axiom.

Table 7: List of all constraints on actions for the SR-GAP transition system. The notation S_1 is used to denote the elements of S without the first one.

A.1 Longest Derivation Computation

This section details the computation of the longest possible derivations for a sentence of length n . For the sake of simplicity, we ignore unary constituents.

		Number or index of sentences
NEGRA-30	train/dev/test	14669/1833/1833
NEGRA-ALL	train/test/dev	18602/1000/1000
TIGERM15	train/dev/test	40468/5000/5000
TIGERHN08	train/dev	$i \pmod{10} > 1/i \equiv 1 \pmod{10}$
TIGERHN08	train/test	$i \not\equiv 0 \pmod{10}/i \equiv 0 \pmod{10}$

Table 8: Details on the standard splits.

SR-GAP There are n shifts and $n - 1$ binary reductions in a derivation. The longest derivation maximises the number of gap actions, by performing as many gap actions as possible before each binary reductions. When S contains k elements, there are $k - 1$ possible consecutive gap actions. So the longest derivation starts by n shifts, followed by $n - 2$ gap actions, one binary reduction, $n - 3$ gap actions, one binary reduction, and so on.

$$\begin{aligned}
L_{gap}(n) &= n + ((n - 2) + 1) + \dots + 1 \\
&= 1 + 2 + \dots + n \\
&= \frac{n(n - 1)}{2}
\end{aligned}$$

This corresponds to the tree on the left-hand side of Figure 5.

SR-SWAP Using the oracle or Maier (2015), the longest derivation for a sentence of length n consists in maximising the number of swaps before each reduction.⁹

After the first shift, the derivation performs repeatedly $n - i$ shifts, $n - i - 1$ swaps and one reduction, i being the number of shifted terminals before each iteration.

$$\begin{aligned}
L_{swap}(n) &= 1 + \sum_{i=1}^{n-1} ((n - i) + (n - i - 1) + 1) \\
&= 1 + 2 \sum_{i=1}^{n-1} (n - i) \\
&= 1 + 2 \frac{n(n - 1)}{2} \\
&= n^2 - n + 1
\end{aligned}$$

This corresponds to the tree on the right-hand side of Figure 5.

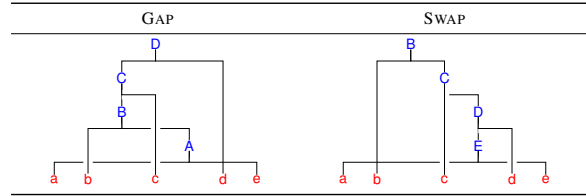


Figure 5: Example tree corresponding to the longest derivation for a sentence of length 5 with GAP and SWAP.

⁹Other possible oracles (Maier and Lichte, 2016) are more efficient on this example and could have different (and better) worst cases.

Neural Architectures for Fine-grained Entity Type Classification

Sonse Shimaoka^{†*}, Pontus Stenetorp[‡], Kentaro Inui[†] and Sebastian Riedel[‡]

[†]Graduate School of Information Sciences, Tohoku University

[‡]Department of Computer Science, University College London

{simaokasonse, inui}@ecei.tohoku.ac.jp

{p.stenetorp, s.riedel}@cs.ucl.ac.uk

Abstract

In this work, we investigate several neural network architectures for fine-grained entity type classification and make three key contributions. Despite being a natural comparison and addition, previous work on attentive neural architectures have not considered hand-crafted features and we combine these with learnt features and establish that they complement each other. Additionally, through quantitative analysis we establish that the attention mechanism learns to attend over syntactic heads and the phrase containing the mention, both of which are known to be strong hand-crafted features for our task. We introduce parameter sharing between labels through a hierarchical encoding method, that in low-dimensional projections show clear clusters for each type hierarchy. Lastly, despite using the same evaluation dataset, the literature frequently compare models trained using different data. We demonstrate that the choice of training data has a drastic impact on performance, which decreases by as much as 9.85% loose micro F1 score for a previously proposed method. Despite this discrepancy, our best model achieves state-of-the-art results with 75.36% loose micro F1 score on the well-established FIGER (GOLD) dataset and we report the best results for models trained using publicly available data for the OntoNotes dataset with 64.93% loose micro F1 score.

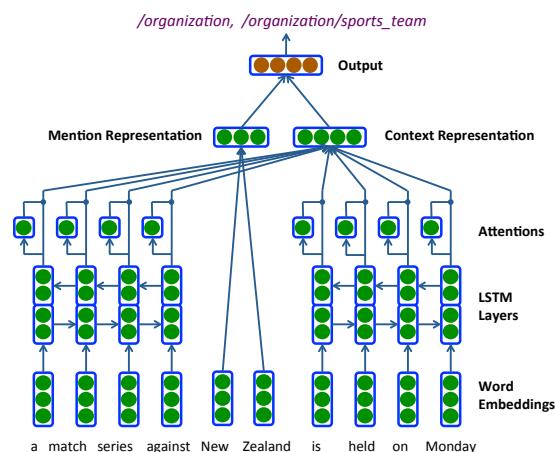


Figure 1: An illustration of the attentive encoder neural model predicting fine-grained semantic types for the mention “New Zealand” in the expression “a match series against New Zealand is held on Monday”.

1 Introduction

Entity type classification aims to label entity mentions in their context with their respective semantic types. Information regarding entity type mentions has proven to be valuable for several natural language processing tasks; such as question answering (Lee et al., 2006), knowledge base population (Carlson et al., 2010), and co-reference resolution (Recasens et al., 2013). A natural extension to traditional entity type classification has been to divide the set of types – which may be too *coarse-grained* for some applications (Sekine, 2008) – into a larger set of *fine-grained* entity types (Lee et al., 2006; Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014; Del Corro et al., 2015); for example *person* into *actor*, *artist*, etc.

Given the recent successes of attentive neural

*This work was conducted during a research visit to University College London.

models for information extraction (Globerson et al., 2016; Shimaoka et al., 2016; Yang et al., 2016), we investigate several variants of an attentive neural model for the task of fine-grained entity classification (e.g. Figure 1). This model category uses a neural attention mechanism – which can be likened to a soft alignment – that enables the model to focus on informative words and phrases. We build upon this line of research and our contributions are three-fold:

1. Despite being a natural comparison and addition, previous work on attentive neural architectures do not consider hand-crafted features. We combine learnt and hand-crafted features and observe that they complement each other. Additionally, we perform extensive analysis of the attention mechanism of our model and establish that the attention mechanism learns to attend over syntactic heads and the tokens prior to and after a mention, both which are known to be highly relevant to successfully classifying a mention.
2. We introduce label parameter sharing using a hierarchical encoding that improves performance on one of our datasets and the low-dimensional projections of the embedded labels form clear coherent clusters.
3. While research on fine-grained entity type classification has settled on using two evaluation datasets, a wide variety of training datasets have been used – the impact of which has not been established. We demonstrate that the choice of training data has a drastic impact on performance, observing performance decreases by as much as 9.85% loose Micro F1 score for a previously proposed method. However, even when comparing to models trained using different datasets we report state-of-the-art results of 75.36% loose micro F1 score on the FIGER (GOLD) dataset.

2 Related Work

Our work primarily draws upon two strains of research, fine-grained entity classification and attention mechanisms for neural models. In this section we introduce both of these research directions.

By expanding a set of coarse-grained types into a set of 147 fine-grained types, Lee et al. (2006)

were the first to address the task of fine-grained entity classification. Their end goal was to use the resulting types in a question answering system and they developed a conditional random field model that they trained and evaluated on a manually annotated Korean dataset to detect and classify entity mentions. Other early work include Sekine (2008), that emphasised the need for having access to a large set of entity types for several NLP applications. The work primarily discussed design issues for fine-grained set of entity types and served as a basis for much of the future work on fine-grained entity classification.

The first work to use distant supervision (Mintz et al., 2009) to induce a large – but noisy – training set and manually label a significantly smaller dataset to evaluate their fine-grained entity classification system, was Ling and Weld (2012) who introduced both a training and evaluation dataset FIGER (GOLD). Arguing that fine-grained sets of types must be organised in a very fine-grained hierarchical taxonomy, Yosef et al. (2012) introduced such a taxonomy covering 505 distinct types. This new set of types lead to improvements on FIGER (GOLD), and they also demonstrated that the fine-grained labels could be used as features to improve coarse-grained entity type classification performance. More recently, continuing this very fine-grained strategy, Del Corro et al. (2015) introduced the most fine-grained entity type classification system to date, covering the more than 16,000 types contained in the WordNet hierarchy.

While initial work largely assumed that mention assignments could be done independently of the mention context, Gillick et al. (2014) introduced the concept of context-dependent fine-grained entity type classification where the types of a mention is constrained to what can be deduced from its context and introduced a new OntoNotes-derived manually annotated evaluation dataset. In addition, they addressed the problem of label noise induced by distant supervision and proposed three label cleaning heuristics. Building upon the noise reduction aspects of this work, Ren et al. (2016) introduced a method to reduce label noise even further, leading to significant performance gains on both the evaluation dataset of Ling and Weld (2012) and Gillick et al. (2014).

Yogatama et al. (2015) proposed to map hand-crafted features and labels to embeddings in or-

der to facilitate information sharing between both related types and features. A pure feature learning approach was proposed by Dong et al. (2015). They defined 22 types and used a two-part neural classifier that used a recurrent neural network to obtain a vector representation of each entity mention and in its second part used a fixed-size window to capture the context of a mention. A recent workshop paper (Shimaoka et al., 2016) introduced an attentive neural model that unlike previous work obtained vector representations for each mention context by composing it using a recurrent neural network and employed an attention mechanism to allow the model to focus on relevant expressions in the mention context. Although not pointed in Shimaoka et al. (2016), the attention mechanism used differs from previous work in that it does not condition the attention. Rather, they used global weights optimised to provide attention for every fine-grained entity type classification decision.

To the best of our knowledge, the first work that utilised an attention architecture within the context of NLP was Bahdanau et al. (2014), that allowed a machine translation decoder to attend over the source sentence. Doing so, they showed that adding the attention mechanism significantly improved their machine translation results as the model was capable of learning to align the source and target sentences. Moreover, in their qualitative analysis, they concluded that the model can correctly align mutually related words and phrases. For the set of neural models proposed by Hermann et al. (2015), attention mechanisms are used to focus on the aspects of a document that help the model answer a question, as well as providing a way to qualitatively analyse the inference process. Rocktäschel et al. (2015) demonstrated that by applying an attention mechanism to a textual entailment model, they could attain state-of-the-art results, as well as analyse how the entailing sentence would align to the entailed sentence.

Our work differs from previous work on fine-grained entity classification in that we use the *same publicly available training data* when comparing models. We also believe that we are the first to consider the *direct combination of hand-crafted features and an attentive neural model*.

Feature	Description	Example
Head	Syntactic head of the mention	Obama
Non-head	Non-head words of the mention	Barack, H.
Cluster	Brown cluster for the head token	1110, ...
Characters	Character trigrams for the mention head	:ob, oba, ...
Shape	Word shape of the mention phrase	Aa A. Aa
Role	Dependency label on the mention head	subj
Context	Words before and after the mention	B:who, A:first
Parent	The head's lexical parent	picked
Topic	The LDA topic of the document	LDA:13

Table 1: Hand-crafted features, based on those of Gillick et al. (2014), used by the sparse feature and hybrid model variants in our experiments. The features are extracted for each entity mention and the example mention used to extract the example features in this table is "... who [Barack H. Obama] first picked ...".

3 Models

In this section we describe the neural model variants used in this paper as well as a strong feature-based baseline from the literature. We pose fine-grained entity classification as a multi-class, multi-label classification problem. Given a mention in a sentence, the classifier predicts the types $t \in \{1, 0\}^K$ where K is the size of the set of types. Across all the models, we compute a probability $y_k \in \mathbb{R}$ for each of the K types using logistic regression. Variations of the models stem from the ways of computing the input to the logistic regression.

At inference time, we enforce the assumption that at least one type is assigned to each mention by first assigning the type with the largest probability. We then assign any additional types based on the condition that their corresponding probabilities must be greater than a threshold of 0.5, which was determined by tuning it using development data.

3.1 Sparse Feature Model

For each entity mention m , we create a binary feature indicator vector $f(m) \in \{0, 1\}^{D_f}$ and feed it to the logistic regression layer. The features used are described in Table 1, which are comparable to those used by Gillick et al. (2014) and Yogatama et al. (2015). It is worth noting that we aimed for this model to resemble the independent classifier model in Gillick et al. (2014) so that it constitutes as a meaningful well-established baseline; however, there are two noteworthy differences. Firstly, we use the more commonly used clustering method of Brown et al. (1992), as opposed to Uszkoreit and Brants (2008), as Gillick

et al. (2014) did not make the data used for their clusters publicly available. Secondly, we learned a set of 15 topics from the OntoNotes dataset using the LDA (Blei et al., 2003) implementation from the popular gensim software package,¹ in contrast to Gillick et al. (2014) that used a supervised topic model trained using an unspecified dataset. Despite these differences, we argue that our set of features is comparable and enables a fair comparison given that the original implementation and some of the data used is not publicly available.

3.2 Neural Models

The neural models from Shimaoka et al. (2016) processes embeddings of the words of the mention and its context; and we adopt the same formalism when introducing these models and our variants. First, the mention representation $v_m \in \mathbb{R}^{D_m \times 1}$ and context representation $v_c \in \mathbb{R}^{D_c \times 1}$ are computed separately. Then, the concatenation of these representations is used to compute the prediction:

$$y = \frac{1}{1 + \exp\left(-W_y \begin{bmatrix} v_m \\ v_c \end{bmatrix}\right)} \quad (1)$$

Where $W_y \in \mathbb{R}^{K \times (D_m + D_c)}$ is the weight matrix.

Let the words in the mention be $m_1, m_2, \dots, m_{|m|}$. Then the representation of the mention is computed as follows:

$$v_m = \frac{1}{|m|} \sum_{i=1}^{|m|} u(m_i) \quad (2)$$

Where u is a mapping from a word to an embedding. This relatively simple method for composing the mention representation is motivated by it being less prone to overfitting.

Next, we describe the three methods from Shimaoka et al. (2016) for computing the context representations; namely, Averaging, LSTM, and Attentive Encoder.

3.2.1 Averaging Encoder

Similarly to the method of computing the mention representation, the Averaging encoder computes the averages of the words in the left and right context. Formally, let l_1, \dots, l_C and r_1, \dots, r_C be the words in the left and right contexts respectively, where C is the window size. Then, for each sequence of words, we compute the average of the

corresponding word embeddings. Those two vectors are then concatenated to form the representation of the context v_c .

3.2.2 LSTM Encoder

For the LSTM Encoder, the left and right contexts are encoded by an LSTM (Hochreiter and Schmidhuber, 1997). The high-level formulation of an LSTM can be written as:

$$h_i, s_i = lstm(u_i, h_{i-1}, s_{i-1}) \quad (3)$$

Where $u_i \in \mathbb{R}^{D_m \times 1}$ is an input embedding, $h_{i-1} \in \mathbb{R}^{D_h \times 1}$ is the previous output, and $s_{i-1} \in \mathbb{R}^{D_h \times 1}$ is the previous cell state.

For the left context, the LSTM is applied to the sequence l_1, \dots, l_C from left to right and produces the outputs $\vec{h}_1^l, \dots, \vec{h}_C^l$. For the right context, the sequence r_C, \dots, r_1 is processed from right to left to produce the outputs $\vec{h}_1^r, \dots, \vec{h}_C^r$. The concatenation of \vec{h}_C^l and \vec{h}_1^r then serves as the context representation v_c .

3.2.3 Attentive Encoder

An attention mechanism aims to encourage the model to focus on salient local information that is relevant for the classification decision. The attention mechanism variant used in this work is defined as follows. First, bi-directional LSTMs (Graves, 2012) are applied for both the right and left context. We denote the output layers of the bi-directional LSTMs as $\vec{h}_1^l, \vec{h}_1^r, \dots, \vec{h}_C^l, \vec{h}_C^r$ and $\overleftarrow{h}_1^l, \overleftarrow{h}_1^r, \dots, \overleftarrow{h}_C^l, \overleftarrow{h}_C^r$.

For each output layer, a scalar value $\tilde{a}_i \in \mathbb{R}$ is computed using a feed forward neural network with the hidden layer $e_i \in \mathbb{R}^{D_a \times 1}$ and weight matrices $W_e \in \mathbb{R}^{D_a \times 2D_h}$ and $W_a \in \mathbb{R}^{1 \times D_a}$:

$$e_i^l = \tanh\left(W_e \begin{bmatrix} \vec{h}_i^l \\ \overleftarrow{h}_i^l \end{bmatrix}\right) \quad (4)$$

$$\tilde{a}_i^l = \exp(W_a e_i^l) \quad (5)$$

Next, the scalar values are normalised such that they sum to 1:

$$a_i^l = \frac{\tilde{a}_i^l}{\sum_{i=1}^C \tilde{a}_i^l + \tilde{a}_i^r} \quad (6)$$

These normalised scalar values $a_i \in \mathbb{R}$ are referred to as attentions. Finally, we compute the sum of the output layers of the bidirectional

¹<http://radimrehurek.com/gensim/>

LSTMs, weighted by the attentions a_i as the representation of the context:

$$v_c = \sum_{i=1}^C a_i^l \begin{bmatrix} \overrightarrow{h_i} \\ \overleftarrow{h_i} \\ \overleftarrow{h_i} \end{bmatrix} + a_i^r \begin{bmatrix} \overrightarrow{h_i} \\ \overleftarrow{h_i} \end{bmatrix} \quad (7)$$

An illustration of the attentive encoder model variant can be found in Figure 1.

3.3 Hybrid Models

To allow model variants to use both human background knowledge through hand-crafted features as well as features learnt from data, we extended the neural models to create new hybrid model variants as follows. Let $v_f \in \mathbb{R}^{D_l \times 1}$ be a low-dimensional projection of the sparse feature $f(m)$:

$$v_f = W_f f(m) \quad (8)$$

Where $W_f \in \mathbb{R}^{D_l \times D_f}$ is a projection matrix. The hybrid model variants are then defined as follows:

$$y = \frac{1}{1 + \exp \left(-W_y \begin{bmatrix} v_m \\ v_c \\ v_f \end{bmatrix} \right)} \quad (9)$$

These models can thus draw upon learnt features through v_m and v_c as well as hand-crafted features using v_f when making classification decisions. While existing work on fine-grained entity type classification have used either sparse, manually designed features or dense, automatically learnt embedding vectors, our work is the first to propose and evaluate a model using the combination of both features.

3.4 Hierarchical Label Encoding

Since the fine-grained types tend to form a forest of type hierarchies (e.g. musician is a subtype of artist, which in turn is a subtype of person), we investigated whether the encoding of each label could utilise this structure to enable parameter sharing. Concretely, we compose the weight matrix W_y for the logistic regression layer as the product of a learnt weight matrix V_y and a constant sparse binary matrix S :

$$W_y^T = V_y S \quad (10)$$

We encode the type hierarchy formed by the set of types in the binary matrix S as follows. Each type is mapped to a unique column in S , where membership at each level of

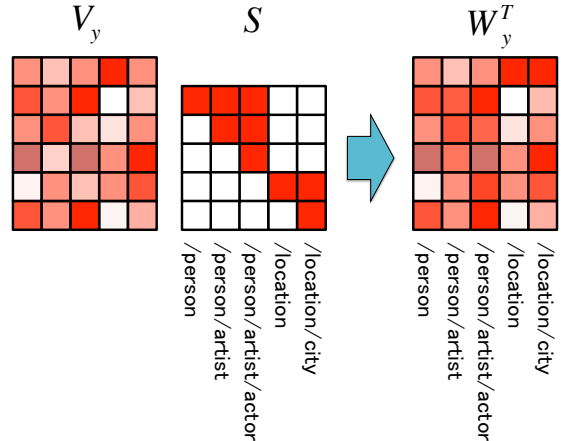


Figure 2: Hierarchical label encoding illustration.

Work	W2M	W2M+D	W2.6M	GN1	GN2
Ling and Weld (2012)	✓				
Gillick et al. (2014)				×	
Yogatama et al. (2015)					×
Ren et al. (2016)	✓	×			
Shimaoka et al. (2016)			✓		

Table 2: Training datasets used and its availability. W2M and W2.6M are Wikipedia-based, +D indicates denoising, and GN1/GN2 are two company-internal Google News datasets. The symbols ✓ and × indicates publicly available and unavailable data.

its type hierarchy is marked by a 1. For example, if we use the set of types defined by Gillick et al. (2014), the column for /person could be encoded as $[1, 0, \dots]$, /person/artist as $[1, 1, 0, \dots]$, and /person/artist/actor as $[1, 1, 1, 0, \dots]$. This encoding scheme is illustrated in Figure 2.

This enables us to share parameters between labels in the same hierarchy, potentially making learning easier for infrequent types that can now draw upon annotations of other types in the same hierarchy.

4 Experiments

4.1 Datasets

Despite the research community having largely settled on using the manually annotated datasets FIGER (GOLD) (Ling and Weld, 2012) and OntoNotes (Gillick et al., 2014) for evaluation, there is still a remarkable difference in the data used to train models (Table 2) that are then evaluated on the same manually annotated datasets. Also worth noting is that some data is not even publicly available, making a

fair comparison between methods even more difficult. For evaluation, in our experiments we use the two well-established manually annotated datasets FIGER (GOLD) and OntoNotes, where like Gillick et al. (2014), we discarded pronominal mentions, resulting in a total of 8,963 mentions. For training, we use the automatically induced publicly available datasets provided by Ren et al. (2016). Ren et al. (2016) aimed to eliminate label noise generated in the process of distant supervision and we use the “raw” noisy data² provided by them for training our models.

4.2 Pre-trained Word Embeddings

We use pre-trained word embeddings that were not updated during training to help the model generalise to words not appearing in the training set (Rocktäschel et al., 2015). For this purpose, we used the freely available 300-dimensional cased word embeddings trained on 840 billion tokens from the Common Crawl supplied by Pennington et al. (2014). For words not present in the pre-trained word embeddings, we use the embedding of the “unk” token.

4.3 Evaluation Criteria

We adopt the same criteria as Ling and Weld (2012), that is, we evaluate the model performance by strict accuracy, loose macro, and loose micro scores.

4.4 Hyperparameter Settings

Values for the hyperparameters were obtained from preliminary experiments by evaluating the model performance on the development sets. Concretely, all neural and hybrid models used the same $D_m = 300$ -dimensional word embeddings, the hidden-size of the LSTM was set to $D_h = 100$, the hidden-layer size of the attention module was set to $D_a = 100$, and the size of the low-dimensional projection of the sparse features was set to $D_l = 50$. We used Adam (Kingma and Ba, 2014) as our optimisation method with a learning rate of 0.001, a mini-batch size of 1,000, and iterated over the training data for five epochs. As a regularizer we

² Although Ren et al. (2016) provided both “raw” data and code to “denoise” the data, we were unable to replicate the performance benefits reported in their work after running their pipeline. We have contacted them regarding this as we would be interested in comparing the benefit of their denoising algorithm for each model, but at the time of writing we have not yet received a response.

Model	Acc.	Macro	Micro
Hand-crafted	51.33	71.91	68.78
Averaging	46.36	71.03	65.31
Averaging + Hand-crafted	52.58	72.33	70.04
LSTM	55.60	75.15	71.73
LSTM + Hand-crafted	57.02	76.98	73.94
Attentive	54.53	74.76	71.58
Attentive + Hand-crafted	59.68	78.97	75.36
FIGER (Ling and Weld, 2012)	52.30	69.90	69.30
FIGER (Ren et al., 2016)	47.4	69.2	65.5

Table 3: Performance on FIGER (GOLD) for models using the *same* W2M training data.

Model	Data	Acc.	Macro	Micro
Attentive + Hand-crafted	W2M	59.68	78.97	75.36
Attentive (Shimaoka et al., 2016)	W2.6M	58.97	77.96	74.94
FIGER + PLE (Ren et al., 2016)	W2M+D	59.9	76.3	74.9
HYENA + PLE (Ren et al., 2016)	W2M+D	54.2	69.5	68.1
K-WASABIE (Yogatama et al., 2015)	GN2	n/a	n/a	72.25

Table 4: Performance on FIGER (GOLD) for models using *different* training data.

used dropout (Hinton et al., 2012) with probability 0.5 applied to the mention representation and sparse feature representation. The context window size was set to $C = 10$ and if the length of a context extends beyond the sentence length, we used a padding symbol in-place of a word. After training, we picked the best model on the development set as our final model and report their performance on the test sets. Our model implementation was done in Python using the TensorFlow (Abadi et al., 2015) machine learning library.

4.5 Results

When presenting our results, it should be noted that we aim to make a clear separation between results from models trained using different datasets.

4.5.1 FIGER (GOLD)

We first analyse the results on FIGER (GOLD) (Tables 3 and 4). The performance of the baseline model that uses the sparse hand-crafted features is relatively close to that of the FIGER system of Ling and Weld (2012). This is consistent with the fact that both systems use linear classifiers, similar sets of features, and training data of the same size and domain.

Looking at the results of neural models, we observe a consistent pattern that adding hand-crafted features boost performance significantly, indicating that the learnt and hand-crafted features complement each other. The effects of adding the hier-

Model	Acc.	Macro	Micro
Hand-crafted	48.16	66.33	60.16
Averaging	46.17	65.26	58.25
Averaging + Hier	47.15	65.53	58.25
Averaging + Hand-crafted	51.57	70.61	64.24
Averaging + Hand-crafted + Hier	51.74	70.98	64.91
LSTM	49.20	66.72	60.52
LSTM + Hier	48.96	66.51	60.70
LSTM + Hand-crafted	48.58	68.54	62.89
LSTM + Hand-crafted + Hier	50.42	69.99	64.57
Attentive	50.32	67.95	61.65
Attentive + Hier	51.10	68.19	61.57
Attentive + Hand-crafted	49.54	69.04	63.55
Attentive + Hand-crafted + Hier	50.89	70.80	64.93
FIGER (Ren et al., 2016)	36.90	57.80	51.60

Table 5: Performance on OntoNotes for models using the *same* W2M training data.

archical label encoding were inconsistent, sometimes increasing, sometimes decreasing performance. We thus opted not to include them in the results table due to space constraints and hypothesise that given the size of the training data, parameter sharing may not yield any large performance benefits. Among the neural models, we see that the averaging encoder perform considerably worse than the others. Both the LSTM and attentive encoder show strong results and the attentive encoder with hand-crafted features achieves the best performance among all the models we investigated.

When comparing our best model to previously introduced models trained using different training data, we find that we achieve state-of-the-art results both in terms of loose macro and micro scores. The closest competitor, FIGER + PLE (Ren et al., 2016), achieves higher accuracy at the expense of lower F1 scores, we suspect that this is due to an accuracy focus in their label pruning strategy. It is worth noting that we achieve state-of-the-art results without the need for any noise reduction strategies. Also, even with 600,000 fewer training examples, our variant with hand-crafted features of the attentive model from Shimaoka et al. (2016) outperforms its feature-learning variant.

In regards to the impact of the choice of training set, we observe that the model introduced in Shimaoka et al. (2016) drops as much as 3.36 points of loose micro score when using a smaller dataset. Thus casting doubts upon the comparability of results of fine-grained entity classification models using different training data.

4.5.2 OntoNotes

Secondly, we discuss the results on OntoNotes (Tables 5, and 6). Again, we see consistent per-

Model	Data	Acc.	Macro	Micro
Averaging + Hand-crafted + Hier	W2M	51.74	70.98	64.91
Attentive + Hand-crafted + Hier	W2M	50.89	70.80	64.93
FIGER + PLE (Ren et al., 2016)	W2M+D	57.2	71.5	66.1
HYENA + PLE (Ren et al., 2016)	W2M+D	54.6	69.2	62.5
Hand-crafted (Gillick et al., 2014)	GN1	n/a	n/a	70.01
K-WASABIE (Yogatama et al., 2015)	GN2	n/a	n/a	72.98

Table 6: Performance on OntoNotes for models using *different* training data.

formance improvements when the sparse hand-crafted features are added to the neural models. In the absence of hand-crafted features, the averaging encoder suffer relatively poor performance and the attentive encoder achieves the best performance. However, when the hand-crafted features are added, a significant improvement occurs for the averaging encoder, making the performance of the three neural models much alike. We speculate that some of the hand-crafted features such as the dependency role and parent word of the head noun, provide crucial information for the task that cannot be captured by the plain averaging model, but can be learnt if an attention mechanism is present. Another speculative reason is that because the training dataset is noisy compared to FIGER (GOLD) (since FIGER (GOLD) uses anchors to detect entities whereas OntoNotes uses an external tool), and the size of the dataset is small, the robustness of the simpler averaging model becomes clearer when combined with the hand-crafted features.

Another interesting observation can be seen for models with the hierarchical label encoding, where it is clear that consistent performance increases occur. This can be explained by the fact that the type ontology used in OntoNotes is more well-formed than its FIGER counterpart. While we do not obtain state-of-the-art performance when considering models using different training data, we do note that in terms of F1-score we perform within 1 point of the state of the art. This being achieved despite having trained our models on different non-proprietary noisy data.

Once again we have an opportunity to study the impact of the choice of training data by comparing the results of the hand-crafted features of Gillick et al. (2014) to our own comparable set of features. What we find is that the performance drop is very dramatic, 9.85 points of loose micro score. Given that the training data for the previously introduced model is not publicly available, we hesi-

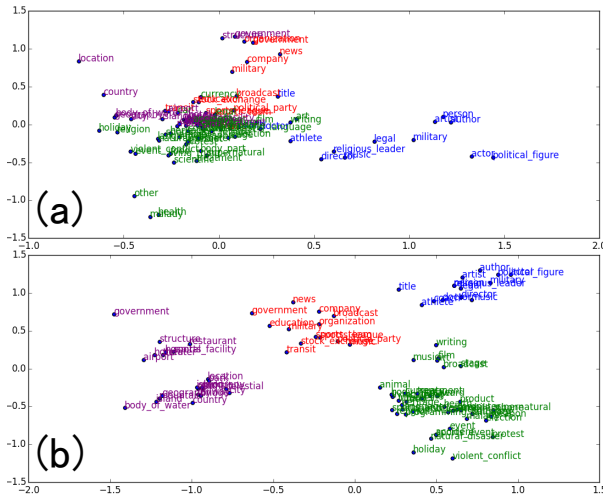


Figure 3: PCA projections of the label embeddings learnt from the OntoNotes dataset where subtypes share the same color as their parent type. Sub-figure (a) uses the non-hierarchical encoding, while sub-figure (b) uses the hierarchical encoding.

tate to speculate as to exactly why this drop is so dramatic, but similar observations have been made for entity linking (Ling et al., 2015). This clearly underlines how essential it is to compare models on an equal footing using the same training data.

4.6 PCA visualisation of label embeddings

By visualising the learnt label embeddings (Figure 3) and comparing the non-hierarchical and hierarchical label encodings, we can observe that the hierarchical encoding forms clear distinct clusters.

4.7 Attention Analysis

While visualising the attention weights for specific examples have become commonplace, it is still not clear exactly what syntactic and semantic patterns that are learnt by the attention mechanism. To better understand this, we first qualitatively analysed a large set of attention visualisations and observed that head words and the words contained in the phrase forming the mention tended to receive the highest level of attention. In order to quantify this notion, we calculated how frequently the word strongest attended over for all mentions of a specific type was the syntactic head or the words before and after the mention in its phrase. What we found through our analysis (Table 7) was that our attentive model without hand-crafted features does indeed learn that head words and the phrase surrounding the mention are highly indica-

Type	Parent	Before	After	Frequent Words
/location	0.319	0.228	0.070	in, at, born
/organization	0.324	0.178	0.119	at, the, by
/art/film	0.207	0.429	0.021	film, films, in
/music	0.259	0.116	0.018	album, song, single
/award	0.583	0.292	0.083	won, a, received
/event	0.310	0.188	0.089	in, during, at

Table 7: Quantitative attention analysis.

tive of the mention type, without any explicit supervision. Furthermore, we believe that this in part might explain why the performance benefit of adding hand-crafted features was smaller for the attentive model compared to our other two neural variants.

5 Conclusions and Future Work

In this paper, we investigated several model variants for the task of fine-grained entity type classification. The experiments clearly demonstrated that the choice of training data – which until now been ignored for our task – has a significant impact on performance. Our best model achieved state-of-the-art results with 75.36% loose micro F1 score on FIGER (GOLD) despite being compared to models trained using larger datasets and we were able to report the best results for any model trained using publicly available data for OntoNotes with 64.93% loose micro F1 score. The analysis of the behaviour of the attention mechanism demonstrated that it can successfully learn to attend over expressions that are important for the classification of fine-grained types. It is our hope that our observations can inspire further research into the limitations of what linguistic phenomena attentive models can learn and how they can be improved.

As future work, we see the re-implementation of more methods from the literature as a desirable target, so that they can be evaluated after utilising the same training data. Additionally, we would like to explore alternative hierarchical label encodings that may lead to more consistent performance benefits.

To ease the reproducibility of our work, we make our code used for the experiments available at <https://github.com/shimaokasonse/NFGEC>.

Acknowledgments

This work was supported by CREST-JST, JSPS KAKENHI Grant Number 15H01702, a Marie

Curie Career Integration Award, and an Allen Distinguished Investigator Award. We would like to thank Dan Gillick for answering several questions related to his 2014 paper.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.
- P.F. Brown, P.V. Desouza, R.L. Mercer, V.J. Della Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.
- Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.
- Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Conference on Empirical Methods in Natural Language Processing*, pages 868–878. ACL.
- Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1243–1249. AAAI Press.
- Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.
- Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, Berlin, Germany, August. Association for Computational Linguistics.
- Alex Graves. 2012. *Supervised sequence labelling*. Springer.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang. 2006. Fine-grained named entity recognition using conditional random fields for question answering. In *Information Retrieval Technology*, pages 581–587. Springer.
- Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *In Proc. of the 26th AAAI Conference on Artificial Intelligence*. Citeseer.
- Xiao Ling, Sameer Singh, and Daniel Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.
- Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.
- Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 627–633, Atlanta, Georgia, June. Association for Computational Linguistics.

- Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. *arXiv preprint arXiv:1602.05307*.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.
- Satoshi Sekine. 2008. Extended named entity ontology with attribute information. In *LREC*, pages 52–57.
- Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. *arXiv preprint arXiv:1604.05525*.
- Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*, pages 755–762, Columbus, Ohio, June. Association for Computational Linguistics.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June. Association for Computational Linguistics.
- Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*, pages 26–31.
- Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *24th International Conference on Computational Linguistics*, pages 1361–1370. ACL.

Author Index

- Abhishek, Abhishek, 797
Abualhaija, Sallam, 870
Adams, Oliver, 937
Adel, Heike, 22, 1183
Agić, Željko, 230
Ajjour, Yamen, 1117
Al Nabki, Mhd Wesam, 35
Alegre, Enrique, 35
Aluísio, Sandra, 315
Anand, Ashish, 797
Anand, Pranav, 742
Ananiadou, Sophia, 991
Ann, Lawrence, 1128
Arevalo, John, 1217
Argueta, Arturo, 1044
Aussenac-Gilles, Nathalie, 262
Awekar, Amit, 797
- Ballesteros, Miguel, 1249
Bar-Haim, Roy, 251
Barrault, Loïc, 1107
Baskaya, Osman, 210
Bekki, Daisuke, 710
Belanger, David, 132
Benton, Adrian, 152
Bergmanis, Toms, 337
Bhattacharya, Indrajit, 251
Bhattacharyya, Pushpak, 818, 1159
Biemann, Chris, 86, 590
Bilu, Yonatan, 176
Bird, Steven, 894, 937
Blanco, Eduardo, 860
Bod, Rens, 1228
Bogdanova, Dasha, 121
Bojar, Ondřej, 927
Bosco, Cristina, 262
Braud, Chloé, 292
Brockmeier, Austin J., 991
Butnaru, Andrei, 916
- Camacho-Collados, Jose, 99
Carman, Mark, 408
Castro Ferreira, Thiago, 655
Catley, Delwyn, 1128
- Chang, Angel, 460
Chang, Ming-Wei, 494
Chatterjee, Rajen, 525
Chen, Wei-Te, 558
Cheng, Jianpeng, 665
Chiang, David, 1044
Chisholm, Andrew, 633
Cho, Kyunghyun, 1053
Cimiano, Philipp, 54
Coavoux, Maximin, 292, 1259
Cohen, Shay B., 536
Cohn, Trevor, 894, 937
Conneau, Alexis, 1107
Cotterell, Ryan, 514
Crabbé, Benoit, 1259
- Damonte, Marco, 536
Das, Amitava, 731
Das, Pradipto, 969
Das, Rajarshi, 132
Datta, Ankur, 969
de Paz, Ivan, 35
Dehouck, Mathieu, 241
Deleris, Lea, 839
Demberg, Vera, 281, 958
Denis, Pascal, 241, 775
Di Fabrizio, Giuseppe, 969
Dinuzzo, Francesco, 251
Dogruoz, A. Seza, 210
Dong, Li, 623
Du, Mian, 1096
Duong, Long, 894
Dyer, Chris, 1249
Dzendszik, Daria, 121
- Eckle-Kohler, Judith, 870
Ekbal, Asif, 1159
Erdem, Aykut, 199
Erdem, Erkut, 199
Eren, Mustafa Tolga, 210
Escoter, Llorenc, 1096
- Fang, Michael, 460
Farah, Benamara, 262
Faralli, Stefano, 86, 590

Farkas, Richárd, 356
Farra, Noura, 1002
Fersini, Elisabetta, 273
Fidalgo, Eduardo, 35
Fokkens, Antske, 677
Foster, Jennifer, 121
Freedman, Marjorie, 601
Futrell, Richard, 688

Gambäck, Björn, 731
Gasic, Milica, 438
Gebremelak, Gebremedhen, 525
Gildea, Daniel, 366
Goggin, Kathy J., 1128
Goldberg, Yoav, 765
Goldwater, Sharon, 337, 1239
González, Fabio A., 1217
Goulermas, John Y., 991
Gruhl, Daniel, 142
Gu, Jiatao, 1053
Guo, Shangmin, 111
Gupta, Samiksha, 731
Gurevych, Iryna, 471, 870, 980

Hachey, Ben, 633
Haffari, Gholamreza, 408, 428
Hartmann, Silvana, 471
Hartung, Matthias, 54
He, Shizhu, 111
He, Yulan, 808
Heigold, Georg, 505
Heinzerling, Benjamin, 828
Heyman, Geert, 1085
Hirao, Tsutomu, 386
Hirst, Graeme, 176
Hockenmaier, Julia, 721
Hou, Yufang, 176
Hough, Julian, 326
Hovy, Dirk, 152
Howcroft, David M., 958
Hristea, Florentina, 916
Huang, Shaohan, 623

Ikizler-Cinbis, Nazli, 199
Inui, Kentaro, 1271
Ionescu, Radu Tudor, 916

Jäger, Gerhard, 1205
Jamatia, Anupam, 731
Jebbara, Soufian, 54
Jin, Gongye, 568
Jochim, Charles, 839
Joshi, Sachindra, 376

Ju, Y.C., 450
Jurafsky, Dan, 460

Kanayama, Hiroshi, 894
Kann, Katharina, 514
Karoui, Jihen, 262
Katinskaia, Anisia, 1096
Kaupmann, Fabian, 54
Kawahara, Daisuke, 568
Kawasaki, Yoshifumi, 1195
Khapra, Mitesh M., 376
Kiela, Douwe, 163
Kilickaya, Mert, 199
Kocmi, Tom, 927
Kong, Lingpeng, 1249
Kontonatsios, Georgios, 991
Korhonen, Anna, 163
Krahmer, Emiel, 655
Kübler, Sandra, 347
Kumar, Upendra, 731
Kuncoro, Adhiguna, 1249
Kurohashi, Sadao, 568
Kuznetsov, Iliia, 471

Lai, Alice, 721
Lan, Man, 1033
Lapata, Mirella, 623, 665, 881
Le, Minh, 677
Lecun, Yann, 1107
Leeuwenberg, Artuur, 1150
Levine, Aaron, 969
Levy, Omer, 765
Levy, Roger, 688
Li, Victor O.K., 1053
Liakata, Maria, 483
Lin, Chin-Yew, 828
List, Johann-Mattis, 1205
Liu, Fei, 1, 305, 754
Liu, Kang, 111
Liu, Qun, 121
Lukin, Stephanie, 742

Ma, Tengfei, 894
Maharjan, Suraj, 1217
Maheshwari, Tushar, 731
Makarucha, Adam, 937
Mallinson, Jonathan, 881
Martin, Teresa, 471
Martínez Alonso, Héctor, 44, 230
Martínez, Paloma, 1014
Martínez-Gómez, Pascual, 710
Mascarell, Laura, 948

McCallum, Andrew, 132, 613
McKeown, Kathy, 1002
Meltzer, Talya, 601
Mendes, Pablo, 142
Messina, Enza, 273
Mihalcea, Rada, 1128
Miller, Tristan, 870
Min, Bonan, 601
Mineshima, Koji, 710
Mirkin, Shachar, 1074
Mitchell, Margaret, 152
Miyao, Yusuke, 710
Moens, Marie-Francine, 1085, 1150
Montes, Manuel, 1217
Mooney, Raymond J., 547
Moriceau, Véronique, 262
Mousa, Amr, 1023
Mrkšić, Nikola, 438
Mu, Tingting, 991
Mukherjee, Atreyee, 347
Munkhdalai, Tsendsuren, 11, 397
Murray, Iain, 1239
Muzny, Grace, 460

Naderi, Nona, 176
Nagata, Masaaki, 386
Nagata, Ryo, 1195
Navigli, Roberto, 99
Neelakantan, Arvind, 132, 613
Negri, Matteo, 525
Neubig, Graham, 937, 1053, 1249
Neumann, Guenter, 505
Nguyen, Kim Anh, 76
Nguyen, Minh Le, 905
Nishino, Masaaki, 386
Nowson, Scott, 754
Nozza, Debora, 273

Padmakumar, Aishwarya, 547
Palmer, Martha, 558
Palshikar, Girish, 818
Panchenko, Alexander, 86, 590
Patel, Raj Nath, 1074
Patti, Viviana, 262
Pawar, Sachin, 818
Pelemans, Joris, 417
Peng, Baolin, 450
Peng, Xiaochang, 366
Perez, Julien, 1, 305, 754
Pérez-Rosas, Verónica, 1128
Pivovarova, Lidia, 1096
Plank, Barbara, 44, 230

Ponzetto, Simone Paolo, 86, 590
Poon, Hoifung, 1171
Popescu-Belis, Andrei, 948
Prabhakaran, Vinodkumar, 176
Procter, Rob, 483
Pu, Xiao, 948
Puduppully, Ratish, 643

Quirk, Chris, 1171
Quiros, Antonio, 1014

Rabinovich, Ella, 1074
Radford, Will, 633
Raganato, Alessandro, 99
Raghu, Dinesh, 376
Ralaivola, Liva, 775
Reddy, Sathish, 376
Reganti, Aishwarya N., 731
Resnicow, Kenneth, 1128
Riedel, Sebastian, 1271
Rojas Barahona, Lina M., 438
Ruppert, Eugen, 86
Rutherford, Attapol, 281

Saha, Amrita, 251
Saha, Sriparna, 1159
Sánchez-Cartagena, Víctor M., 1063
Santus, Enrico, 65
Sarabi, Zahra, 860
Sato, Motoki, 991
Satta, Giorgio, 536
Scheutz, Matthias, 347
Schlangen, David, 326
Schlechtweg, Dominik, 65
Schlichtkrull, Michael, 220
Schuller, Björn, 1023
Schulte im Walde, Sabine, 76
Schütze, Hinrich, 22, 514, 578, 699, 785, 1183
Schwenk, Holger, 1107
Segura-Bedmar, Isabel, 1014
Seltzer, Michael, 450
Sennrich, Rico, 881
Søgaard, Anders, 220, 230, 292, 765
Shimaoka, Sonse, 1271
Shoemark, Philippa, 1239
Shrimpton, Luke, 1239
Shrivastava, Manish, 643
Shroff, Gautam, 850
Shulby, Christopher, 315
Shwartz, Vered, 65
Silva de Carvalho, Danilo, 905
Simkó, Katalin, 356

Singh, Satinder, 1128
Slonim, Noam, 251
Smith, Noah A., 1249
Sofroniev, Pavel, 1205
Solorio, Thamar, 1217
Specia, Lucia, 1074
Stab, Christian, 980
Stamatatos, Efstathios, 1138
Stanovsky, Gabriel, 142
Stein, Benno, 176, 1117
Stenetorp, Pontus, 1271
Strube, Michael, 828
Su, Pei-Hao, 438
Sun, Changzhi, 1033
Sun, Shiliang, 1033
Sur, Debnil, 1239
Suzuki, Jun, 386
Szántó, Zsolt, 356

Takamura, Hiroya, 1195
Thijm, Tim Alberdingk, 176
Thomason, Jesse, 547
Toral, Antonio, 1063
Tran, Quan Hung, 428
Tran, Tuan Dung, 408
Treviso, Marcos, 315
Tsuji, Jun'ichi, 991
Tuggener, Don, 188
Tunaoglu, Doruk, 210
Turchi, Marco, 525

Ultes, Stefan, 438
Upadhyay, Shyam, 494

van Cranenburgh, Andreas, 1228
van Genabith, Josef, 505
Van hamme, Hugo, 417
Vandyke, David, 438
Verga, Patrick, 613
Verwimp, Lyan, 417
Vig, Lovekesh, 850
Vincze, Veronika, 356
Vu, Ngoc Thang, 76
Vulić, Ivan, 163, 1085

Wachsmuth, Henning, 176, 1117
Walker, Marilyn, 742
Wambacq, Patrick, 417
Wang, Bo, 483
Wang, Chuan, 366
Wei, Furu, 623
Wen, Tsung-Hsien, 438
Whittaker, Steve, 742

Wintner, Shuly, 1074
Wong, Kam-Fai, 450
Wu, Yuanbin, 1033
Wubben, Sander, 655

Xia, Yandi, 969
Xu, Ke, 623
Xue, Nianwen, 281, 366

Yadav, Mohit, 850
Yadav, Shweta, 1159
Yaghoobzadeh, Yadollah, 578, 1183
Yangarber, Roman, 1096
Yildiz, Eray, 210
Yin, Wenpeng, 699
Young, Steve, 438
Yu, Hong, 11, 397

Zeng, Xiangrong, 111
Zhang, Qi, 1033
Zhang, Xingxing, 665
Zhang, Xuan, 808
Zhang, Yue, 643
Zhao, Jun, 111
Zhou, Deyu, 808
Zhou, Ming, 623
Zimmermann, Karl-Heinz, 870
Zubiaga, Arkaitz, 483
Zukerman, Ingrid, 428
Zweig, Geoffrey, 450